Univerza v Ljubljani
Fakulteta za računalništvo in informatiko

Jurij Mihelič

Prilagodljivost v optimizacijskih problemih

DOKTORSKA DISERTACIJA

Mentor: prof. dr. Borut Robič

Ljubljana, 2006

Zahvala

Videl sem dlje, ker sem stal na ramenih velikanov.
– Isaac Newton

Najprej se iskreno zahvaljujem svojemu mentorju prof. dr. Borutu Robiču za vse predloge, spodbude in ideje tako v času pripravljanja disertacije kot tudi med samim podiplomskim študijem. Posebej se mu zahvaljujem, da mi je po končanem dodiplomskem študiju v svojem scenariju predvidel atribut mladega raziskovalca. Kadar sem imel občutek, da sem se znašel v nedoločenosti, mi je bil vedno prilagodljivo na razpolago.

Prve zamisli o prilagodljivosti so se rodile iz pisanja enačb in risanja grafov po šolskih tablah v Ljubljani in Grenoblu. Pri tem sta mi optimalno krajšala čas dr. Amine Mahjoub in dr. Christophe Rapine, za kar sem jima posebej hvaležen.

Za podporo pri raziskovanju se zahvaljujem celotni zasedbi Laboratorija za algoritme in podatkovne strukture, tako nekdanjemu predstojniku laboratorija prof. dr. Boštjanu Vilfanu kot Urošu Čibeju, doc. dr. Tomažu Dobravcu in doc. dr. Boštjanu Slivniku.

Za slovnični in stilistični pregled besedila se zahvaljujem Tadeji Rode, ki je bila v zadnjih trenutkih pisanja pogosto primorana prilagoditi svoj že tako polinomski prosti čas.

Za svobodo pri sprejemanju življenjskih odločitev se zahvaljujem svojima prilagodljivima staršema. Hvala mami Julijani za stohastično izražanje skepticizma do mojega nočnega strmenja v monitor in hvala očetu Viktorju, ki je v večni nedoločenosti že zdavnaj rešil problem prilagodljivih atributov. Hvala sestrama Klavdiji in Lidiji ter njunima družinama za kvantitativne nasvete in kvalitativna navodila. Hvala Tini, ki je zame najbolj prilagodljiva oseba na svetu, ker mi je pomagala rešiti problem največjega ujemanja. Hvala tudi Tolsti gori, drevesom in zvezdam za deterministično odvajanje pretoka negativne energije.

Hkrati se zahvaljujem takratnemu Ministrstvu za šolstvo, znanost in šport Republike Slovenije, ki mi je odobrilo status mladega raziskovalca, in s tem poskrbelo za finančno podporo v času študija. In nenazadnje se zahvaljujem vsem statičnim prijateljem, dinamičnim znancem in drugim, ki jih v zahvali nisem izrecno omenil, čeprav so tudi oni posredno ali neposredno pripomogli k nastanku tega dela.

Povzetek

Doktorska disertacija *Prilagodljivost v optimizacijskih problemih* raziskuje optimizacijske probleme, v katerih je potrebno poiskati več različnih rešitev, ki so si med seboj na nek način podobne. Takšnim problemom pravimo tudi prilagodljivostni problemi in njihovim rešitvam prilagodljive rešitve.

Različni optimizacijski problemi se pogosto pojavljajo v praksi, zato je njihovo proučevanje zelo pomembno tako s praktičnega stališča kakor tudi za razvoj znanosti. V praksi se rado primeri, da je v vhodnih podatkih optimizacijskih problemov prisotna *nedoločenost*. Podatek je *nedoločen*, če ne moremo povedati, kakšna je njegova natančna vrednost. V literaturi se pojavlja več pristopov za obvladovanje nedoločenosti podatkov, kot sta npr. robustnost in stohastičnost. Izkaže se, da je prilagodljivost med drugim uporabna tudi kot način obvladovanja nedoločenosti vhodnih podatkov v optimizacijskih problemih.

Eden izmed načinov modeliranja nedoločenosti podatkov je modeliranje s *scenariji*. Scenarij v splošnem pravzaprav predstavlja nalogo nekega optimizacijskega problema, medtem ko je naloga prilagodljivostnega optimizacijskega problema nabor scenarijev. Med scenariji pogosto ne dovolimo poljubnega razlikovanja, v ta namen lahko uporabimo metodo *perturbiranja* scenarijev.

Prilagodljivostnih optimizacijskih problemov se lotimo predvsem z vidika teorije računske zahtevnosti in analize algoritmov. Pri tem uporabljamo pojme, kot so Karpove in Turingove prevedbe, razredi zahtevnosti odločitvenih in optimizacijskih problemov, hevristični in aproksimacijski algoritmi itd. Poleg tega pri definiciji in reševanju problemov največ uporabljamo pojme iz teorije grafov, še posebej s področja problemov pretokov v omrežju in iskanja ujemanj v grafu.

Prilagodljivost je zelo splošen pojem, ki se na takšen ali drugačen način že pojavlja v nekaterih optimizacijskih problemih, kot so npr. problemi s področja realnočasovnih sistemov, online (sprotnih) algoritmov, dinamičnosti podatkov, mobilnosti ponudnikov v problemih razmeščanja itd. Prilagodljivost in z njo povezane pojme zato najprej natančno definiramo, nato pa opišemo način, kako lahko prilagodljivost vpeljemo v že znane optimizacijske probleme.

Nalogo prilagodljivostnega optimizacijskega problema sestavlja *graf* $G = (V, E)$ *prehodov scenarijev*, pri čemer vsakemu vozlišču $i \in V$ pripada scenarij s_i, ki opisuje nek nabor vhodnih podatkov. Za vsak scenarij s_i je potrebno poiskati *partikularno rešitev* S_i, tako da je njena kakovost $z_i(S_i)$ optimalna, ali v primeru $\mathcal{NP}$-težkih pro-

blemov približna. Poleg tega je podan še način izračuna *partikularne prilagodljivosti* $f_{ij}(S_i, S_j)$ za vse $(i, j) \in E$, ki predstavlja medsebojno podobnost dveh rešitev. Kriterijska funkcija problema je sestavljena iz partikularnih prilagodljivosti, npr. kot $\max_{(i,j)\in E} f_{ij}$. V problemih prilagajanja gre navadno za minimizacijo. Na ta način pravzaprav definiramo več podobnih si vrst prilagodljivosti.

Glavni del disertacije je posvečen problemom prilagodljivih atributov. V teh problemih se osredotočimo na optimizacijo *celotne prilagodljivosti* in na enostavno predstavitev scenarijev, ki jih opišemo z množico *atributov*. Definiramo več različic teh problemov, za katere obdelamo zahtevnost reševanja, t.j. dokažemo $\mathcal{NP}$-težkost, ter opišemo njihovo absolutno in relativno aproksimabilnost. Poleg tega obdelamo več različic teh problemov, v katerih se v vsaki osredotočimo na neko vrsto nalog. Tako se lotimo problemov prilagodljivih atributov na drevesu, splošnem grafu, verigi in dvostopenjskem grafu. Za vse naštete predstavimo algoritme za njihovo optimalno ali približno reševanje.

Lotimo se tudi vpeljave prilagodljivosti v nekatere že znane optimizacijske probleme, kot sta npr. razmeščanje centrov in najmanjše vpeto drevo. Za prilagodljive različice teh problemov pokažemo $\mathcal{NP}$-težkost. Poleg tega za vsakega od njih opišemo tudi algoritme za optimalno ali hevristično reševanje.

V zadnjem poglavju naštejemo avtorjeve izvirne prispevke k znanosti in predstavimo nekatera odprta vprašanja ter predloge za nadaljevanje dela.

Ključne besede: optimizacijski problem, prilagodljivost, nedoločenost, scenarij, teorija računske zahtevnosti, hevristični in aproksimacijski algoritem.

Abstract

The thesis *Flexibility in optimization problems* examines optimization problems in which one must find several different solutions which exhibit a certain degree of mutual resemblance. Such problems are called flexibility problems while their solutions are said to be flexible.

There is a multitude of optimization problems in practice. Therefore, their close examination is very important both from practical point of view as well as for scientific progress. However, *uncertainty* of input data is often present in practical optimization problems. A datum is *uncertain* if its exact value is not known. There are several approaches to dealing with uncertainty of input data. In addition to robustness and stochastics, it turns out that the flexibility is also a possible approach to dealing with uncertainty of input data in optimization problems.

The uncertainty of data can be modeled with *scenarios*. In general, a scenario represents an instance of an optimization problem while an instance of a flexibility optimization problem is represented by a set of scenarios. Discrepancies among the scenarios are often bounded. For this purpose, a method of *perturbating* scenarios can be applied.

We tackle various flexibility optimization problems mainly from the point of view of computational complexity theory and algorithm analysis. Hence, notions such as Karp and Turing reducibility, complexity classes of decision or optimization problems, heuristic and approximation algorithms are often used. For defining and solving problems we mainly use notions from graph theory and special problem areas, such as network flows and matching problems. Flexibility is a very general idea and is, therefore, in one way or another already present in some optimization problems, for example in real-time systems, online algorithms, dynamic data, mobility of facilities in location problems etc. For this reason, we formally define the notion of flexibility and other related notions, and subsequently describe a general method of introducing the flexibility into optimization problems.

An instance of a flexibility optimization problem consists of a *graph* $G = (V, E)$ *of scenario transitions* where to each $i \in V$ a scenario s_i is assigned, describing some set of input data. For each scenario s_i the *particular solution* S_i is found in such a way that its quality $z_i(S_i)$ is either optimal or approximate in case of an $\mathcal{NP}$-hard problem. Furthermore, a way of calculating the *particular flexibility* $f_{ij}(S_i, S_j)$ for all $(i, j) \in E$ is assumed to be given. The objective function of the

problem is composed of these particular flexibilities, such as $\max_{(i,j)\in E} f_{ij}$. The goal of flexibility problems is usually minimization. Other kinds of flexibility problems are similarly defined.

The main part of the thesis is focused on various flexible-attribute problems. Here we study the optimization of *complete flexibility* and concentrate on simple description of scenarios via sets of *attributes*. We define several versions of the problem. We examine their computational complexity in terms of time consumption, we prove their $\mathcal{NP}$-hardness, and find positive and negative results about their absolute and relative approximability. Due to their intractability, we treat various simplified versions of the problems. In particular, we analyze flexible-attribute problems on trees, general graphs, chains, and two-stage graphs. For all these problems we design and analyze algorithms for constructing optimal or approximate solutions.

Next, we discuss introducing the flexibility in some familiar optimization problems such as locating centers and the minimum spanning tree. For all flexibility versions of these problems we prove $\mathcal{NP}$-hardness. Furthermore, for all these problems we design either exact or heuristic algorithms.

The final chapter contains original contributions to the science, open questions, and ideas for further research in the field.

Keywords: optimization problem, flexibility, uncertainty, scenario, computational complexity theory, heuristic and approximation algorithm.

Kazalo

Poglavje 1

Uvod

Vsi ljudje po naravi stremijo k védenju.
– Aristotel

V uvodnem poglavju najprej na kratko izpostavimo nekaj pomembnejših razlogov za proučevanje optimizacijskih problemov. V praksi se rado primeri, da je v vhodnih podatkih optimizacijskih problemov prisotna nedoločenost. Zato se z nedoločenostjo seznanimo in naštejemo nekaj pristopov za njeno obvladovanje, med katerimi omenimo tudi prilagodljivostni pristop, ki sestavlja bistveni del disertacije. Zatem predstavimo poglavitne cilje in namen dela. Na koncu poglavja bralca popeljemo še skozi vsebinsko predstavitev disertacije.

1.1 Pomembnost optimizacijskih problemov

Raznovrstni optimizacijski problemi se pojavljajo pri modeliranju praktičnih problemov v mnogih gospodarskih panogah, kot so npr. ekonomija, transport in distribucija, skladiščenje, regijsko in mestno načrtovanje, gradbeništvo, arhitektura, geografija, industrijsko inženirstvo, strojništvo, elektronika, računalništvo in informatika, fizika, kemija, biologija itd. Gre za aplikacije, kot so maksimizacija dobička, minimizacija časa transporta ali skladiščenja, razmeščanje policijskih patrulj, ambulant, gasilskih postaj, različnih komponent strojev, distribucijskih mest, bančnih podružnic, motelov, načrtovanje, sestavljanje in testiranje elektronskih vezij, razvrščanje opravil, letov letal in osebja, kodiranje, dekodiranje, stiskanje in razne druge obdelave podatkov itd. [CC03, Das95, DH02, ES04, GN96, MF98, ReV97, RE05, Ski98].

Temelja učinkovitega reševanja optimizacijskih problemov sta *hitrost*, t.j. čas, potreben za iskanje rešitve, in *kakovost*, t.j. cena najdene rešitve. Oba kriterija, hitrost in kakovost, sta zelo pomembna. Hitrost je pomembna, ker za iskanje rešitve nimamo na voljo neomejeno časa, kakovost zato, ker so drage rešitve pogosto neuporabne. Kakovostna rešitev lahko pripelje do učinkovitejšega izkoriščanja naravnih virov in zmogljivosti, za katere je iz ekonomije dobro znano, da jih ni na voljo v

neomejenih količinah ter da zaradi njih lahko pride celo do razsežnejših mednarodnih sporov.

Oba kriterija sta v praksi pogosto obratno sorazmerna, zato obojestranska zadovoljitev obeh ni vedno izvedljiva. Dobro poznavanje matematičnih modelov, metod in postopkov s področja optimizacijskih problemov torej vodi do njihovega hitrejšega in kakovostnejšega reševanja, boljšega izkoriščanja naravnih virov in zmogljivosti, učinkovitejše obdelave materialov in pridelave dobrin ter nenazadnje morda tudi do zgladitve medsebojnih konfliktov.

Probleme, ki jih je možno hitro in hkrati kakovostno reševati, kar pomeni, da je optimalno rešitev problema mogoče poiskati v polinomskem času, razvrstimo v razred $\mathcal{P}$[1]. Žal pa se veliko praktičnih problemov večkrat upira hitremu in hkrati kakovostnemu reševanju. Izkaže se, da večina učinkovitemu reševanju upirajočih se problemov sodi v razred $\mathcal{NP}$-težkih optimizacijskih problemov. Ti problemi so "težki" v nasprotju z "lahkimi" problemi iz razreda $\mathcal{P}$.

Do danes je znanih že na tisoče $\mathcal{NP}$-težkih optimizacijskih problemov [ACG+99, GJ79]. Za prvega izmed njih, problem izpolnljivosti[2] Boolovih izrazov, je $\mathcal{NP}$-težkost dokazal Stephen Cook [Coo71]. S pomočjo tega problema je Richard Karp uporabil postopek *prevedb* in nadalje dokazal $\mathcal{NP}$-težkost še nekaterih drugih optimizacijskih problemov [Kar72]. S tem se je začel razmah teorije računske zahtevnosti[3] problemov in algoritmov. Sem sodi tudi ena izmed najpomembnejših še nerešenih "ugank"[4] teoretičnega računalništva, t.j. vprašanje enakosti oz. različnosti razredov $\mathcal{P}$ in $\mathcal{NP}$. Kljub izmikanju odgovora večina današnjih raziskovalcev verjame, da $\mathcal{P} \neq \mathcal{NP}$.

1.2 Obvladovanje nedoločenosti in prilagodljivost

Vhodni podatki večine problemov, obdelovanih v znanstveni literaturi, so popolnoma *določeni*; npr. vhodni podatki so deterministični, ne vsebujejo verjetnosti, so zanesljivi in natančno opisujejo praktično situacijo, se ne spreminjajo s časom oz. je njihova časovna odvisnost popolnoma določena itd. Vendar v praksi temu ni vedno tako. Zaradi različnih razlogov se večkrat pojavi nedoločenost, netočnost, negotovost, nezanesljivost ali kakšna druga nejasnost podatkov. V nadaljevanju bomo za vse te oblike nejasnosti podatkov uporabljali pojem njihove *nedoločenosti*.

Seveda vhodni podatki, ki nastopajo v problemih, nikoli niso popolnoma nedoločeni, ampak imamo o njih vedno vsaj neko delno informacijo. (O popolnoma nedoločenih podatkih ne bi vedeli ničesar, kar je enako, kot da ne bi obstajali.) Ko bomo v prihodnje govorili o nedoločenih podatkih, bomo s tem pravzaprav vedno

[1] Opis pojmov $\mathcal{P}$, $\mathcal{NP}$, $\mathcal{NP}$-polnost, $\mathcal{NP}$-težkost itd. sledi v nadaljevanju.

[2] *angl. satisfiabiliy problem.*

[3] *angl. computational complexity theory.*

[4] Za pravilno razrešitev vprašanja je Clayev matematični inštitut leta 2000 razpisal bogato nagrado milijon dolarjev. (http://www.claymath.org)

imeli v mislih njihovo delno nedoločenost oz. določenost.

Pojavljanje nedoločenosti podatkov v praktičnih problemih je poglavitni razlog, da njihovo raziskovanje v zadnjem času precej pridobiva na pomembnosti in pozornosti. Nedoločenost podatkov v problemih obvladujemo z različnimi pristopi, kot so npr. robustna optimizacija, stohastična optimizacija, sprotno reševanje itd. [BEY98, Čib02, KY97].

Eden izmed načinov obvladovanja nedoločenosti je tudi iskanje rešitev s sposobnostjo prilagajanja, t.j. rešitev, ki se lahko spreminjajo in se na ta način lahko prilagodijo različnim vrednostim vhodnih podatkov. *Prilagodljivost* je v optimizacijskih problemih pravzaprav nov pojem, zato je najprej del disertacije posvečen njenemu splošnemu opisu in definiciji, kasneje pa se je lotimo precej bolj podrobno. Kljub neraziskanosti prilagodljivosti pa se na nekaterih področjih in v nekaterih problemih na takšen ali drugačen način že pojavlja [ABE05, EBS05, Hen99, HOH, LLH05, Lin05, LMV05, MG05].

V nadaljevanju se bomo večinoma ukvarjali s prilagodljivostjo v optimizacijskih problemih. Izkaže se, da prilagodljivost ni uporabna samo za obvladovanje nedoločenosti vhodnih podatkov, ampak tudi kot taka, t.j. za modeliranje nekaterih problemov, ki ne vsebujejo nedoločenosti.

1.3 Cilji in namen dela

Glavni namen dela je obdelati področje prilagodljivosti v optimizacijskih problemih. Področje je zelo obširno, saj lahko iz skorajda vsakega znanega optimizacijskega problema izpeljemo njegovo prilagodljivostno različico. Poleg glavnega cilja smo si zato zadali še nekaj podciljev.

Prilagodljivost v optimizacijskih problemih je nov pojem, ki je v literaturi dokaj slabo obdelan, zato se zdi smiselno obravnavo prilagodljivosti pričeti in zaobjeti s čim širšega in čimbolj splošnega vidika ter pri tem paziti na usklajenost novih pojmov z že uveljavljenimi. Potrebno je natančno povedati, kako prilagodljivost vpeljemo v že znane optimizacijske probleme in kakšne so možne različice te vpeljave.

Eden izmed smotrov prilagodljivosti je obvladovanje nedoločenosti vhodnih podatkov, zato bomo namenili pozornost tudi tej. Obdelali bomo različne možnosti pojavljanja nedoločenosti, kakor tudi načine modeliranja nedoločenih podatkov. Naš cilj je tudi razjasniti vrste nedoločenosti podatkov, za katere je prilagodljivostni pristop uporaben in za katere ni.

Poleg splošnega načina vpeljave prilagodljivosti v poljuben optimizacijski problem nas seveda posebej zanima še kak konkreten primer vpeljave. Ker je takih primerov zelo veliko, bomo najprej skušali poiskati tak prilagodljivostni problem, ki bo dovolj enostaven za teoretično obdelavo in nam bo poleg tega služil tudi kot nekakšen "tipičen" predstavnik prilagodljivostnih problemov.

S proučevanjem tega predstavnika želimo pridobiti (neposredno in posredno) čimveč informacij o ostalih prilagodljivostnih problemih. Ker pa samo proučevanje

tipičnega predstavnika seveda ni dovolj, bomo prilagodljivost vpeljali v nekaj že znanih optimizacijskih problemov in jih podrobneje proučili.

Poudarek bo na teoretičnem proučevanju problemov, predvsem z vidika teorije računske zahtevnosti, analize in snovanja algoritmov. Pri tem bomo poskušali obdelane algoritme opisati na način, ki bo omogočal kar najlažjo implementacijo. Pri razvoju algoritmov bomo največ uporabljali naslednja dva pristopa: izpeljava novega algoritma iz kakega že obstoječega algoritma (prilagoditev ali posplošitev znanih postopkov) in razvoj popolnoma novih algoritmov (po kakšni od že znanih metod snovanja algoritmov).

1.4 Predstavitev vsebine

Bistveni del vsebine te disertacije sestavlja proučevanje, vključevanje in vpeljava prilagodljivosti v optimizacijske probleme. Izrazje, pojme in metodologijo bomo črpali predvsem iz področij, kot so: diskretne strukture, teorija množic, teorija grafov, analiza algoritmov, teorija izračunljivosti in zahtevnosti, kombinatorična optimizacija itd. Omenjena področja so v večji meri pokrita tudi v naslednjih preglednih delih [ACG+99, AHU74, CLRS01, GJ79, GKP89, Hoc95, HMU01, KT06, Knu99, KV00, Meh84, Pap94, PS98, Ree95, Rob02, Ski98, Vaz01, Vil02, WW97].

Disertacijo sestavljajo štiri večja poglavja: optimizacijski problemi, prilagodljivost, problemi prilagodljivih atributov in prilagodljivostni optimizacijski problemi. Prvo od naštetih poglavij je pregledno in je namenjeno seznanitvi (ali osvežitvi) z osnovnimi pojmi s področja optimizacijskih problemov, samemu modeliranju optimizacijskih problemov ter njihovemu reševanju. V zadnjem razdelku tega poglavja predstavimo in obdelamo še prisotnost nedoločenosti vhodnih podatkov v optimizacijskih problemih. Bralec, ki mu omenjena področja niso tuja, lahko to poglavje mirne vesti preskoči. Vseeno pa bi morda opozorili vsaj na zadnji razdelek poglavja, ki obravnava nedoločenost, ker je raziskovanje le-te v literaturi dokaj slabo zastopano. Če bi bralec pri obravnavi nadaljnjih poglavij morda le naletel na kakšno nejasnost, naj dodatno razlago poišče v tem poglavju ali v kateri od zgoraj naštetih knjig.

Poglavje o prilagodljivosti je zelo splošno in je namenjeno spoznavanju osnovnih pojmov s področja prilagodljivosti. V poglavju opišemo splošen način vpeljave prilagodljivosti v že obstoječe optimizacijske probleme, poleg tega pa tudi vse nove pojme, ki jih pri tem potrebujemo. Na koncu poglavja predstavimo še različne vrste prilagodljivosti, ki jih razčlenimo predvsem glede na kriterijske funkcije in tipične vhodne podatke, ki se lahko pojavijo v problemu. Celotno poglavje je zelo pomembno za dobro in pregledno razumevanje prilagodljivosti.

V poglavju o problemih prilagodljivih atributov se lotimo tipičnega predstavnika problemov prilagajanja; gre pravzaprav za več različic podobnih si problemov. V vseh različicah se osredotočimo izključno na optimizacijo cene spremembe rešitve pri prehodu iz ene situacije v drugo ter poleg tega skušamo poiskati kar se da splošen in

enostaven opis za modeliranje situacij. Glede na možne različne kriterije definiramo posamezne probleme, nato obdelamo njihovo zahtevnost reševanja, kakor tudi algoritme za njihovo reševanje. V nadaljnjih razdelkih se lotimo različnih poenostavitev ali pristopov k reševanju osnovnih problemov prilagajanja atributov in jih prav tako podrobno obdelamo. Na koncu sledi še del, v katerem prikažemo nekaj potencialnih aplikacij problemov prilagodljivih atributov.

V poglavju o prilagodljivostnih optimizacijskih problemih v vsakem od razdelkov vpeljemo prilagodljivost v nek že znan optimizacijski problem. Za vsak prilagodljivostni optimizacijski problem se najprej lotimo zahtevnosti reševanja. Pri tem pogosto uporabljamo rezultate iz predhodnega poglavja. Obdelamo različne probleme s področja teorije grafov in razmeščanja ponudnikov.

Na koncu sledi zaključek disertacije, v katerem naštejemo poglavitne prispevke k znanosti, kakor tudi odprta vprašanja za nadaljnje raziskovalno delo.

Poglavje 2

Optimizacijski problemi

Nič ni bolj objektivnega od aritmetičnih zakonov.
– Gottlob Frege.

Predstavitev pomembnejših pojmov s področja optimizacijskih problemov je predmet tega poglavja. V prvem razdelku opišemo različne načine nastopanja podatkov v optimizacijskih problemih. Poleg tega opišemo še osnovne pojme s področja teorije grafov. Zatem v drugem razdelku formalno definiramo optimizacijski problem, predstavimo analizo zahtevnosti algoritmov in problemov, opišemo Karpove in Turingove prevedbe ter vpeljemo pomembnejše razrede zahtevnosti problemov. V tretjem razdelku opišemo možnosti, ki se porajajo, kadar se lotimo teoretične obdelave nekega optimizacijskega problema. Poudarek razdelka je predvsem na približnem reševanju problemov, zato definiramo tudi pojem aproksimacijskega algoritma.

V zadnjem razdelku se lotimo obdelave nedoločenosti vhodnih podatkov v optimizacijskih problemih. Nedoločenost poskušamo kar se da natančno opisati in pojasniti vzroke za njeno prisotnost. Zatem se lotimo modeliranja nedoločenosti s scenariji in nazadnje opišemo še obvladovanje nedoločenosti z robustno in stohastično optimizacijo. V disertaciji pogosto uporabljamo pojme in metode s področja problemov pretokov v omrežju. Zato v zadnjem razdelku poglavja opišemo dva različna problema iskanja pretoka v omrežju. Poleg tega pa opišemo še nekatere probleme iskanja ujemanj v grafu.

2.1 Predstavitev in opis podatkov

Poljuben optimizacijski problem opišemo tako, da podamo nalogo, vrsto dopustnih rešitev, kriterijsko funkcijo in cilj optimizacije. Nalogo sestavljajo podatki, ki jih predstavimo z nekim opisnim jezikom; največkrat gre pri tem za kombinacijo matematičnih simbolov in naravnega jezika. V podrobnosti se na tem mestu ne bomo spuščali, vseeno pa bomo predstavili nekatere načine opisovanja podatkov v optimizacijskih problemih. Le ti so pomembni za poglobljeno razumevanje nedoločenosti.

2.1.1 Kvantitativnost in kvalitativnost

Zanimiva in večkrat samoumevna je razčlemba na *kvantitativni* in *kvalitativni* način opisa podatkov.

Kvantitativni podatki. Gre za podatke, ki so opisani na numeričen način, t.j. vsebina podatka je neka številska vrednost. Včasih natančna številska vrednost podatka ni znana. V takem primeru lahko poznamo npr. zalogo vrednosti podatka, t.j. nabor vrednosti, ki jih podatek lahko zavzame. Primeri kvantitativnih podatkov so: cena vozlišča, dolžina povezave, povpraševanje porabnika itd. Zaloge vrednosti kvantitativnih podatkov pa so lahko npr. naravna števila $\mathbb{N}$ ali realna števila $\mathbb{R}$ itd.

Kvalitativni podatki. Ti so nekakšno nasprotje kvantitativno opisanih podatkov. Kvalitativni podatki so torej nenumerični, njihova vsebina navadno predstavlja neko notranjo strukturo oz. sestavo, odnose ali povezave med posameznimi sestavinami in podobno. Kvalitativno opisani podatki so lahko *nesestavljeni*, kot so npr. vozlišča grafa, posli stroja, ponudniki in porabniki storitve itd., ali *sestavljeni* iz enostavnejših podatkov, kot so npr. urejeni pari, množice, permutacije, grafi.

Seveda se v praksi pogosto pojavijo podatki, ki ne sodijo strogo v eno izmed obeh skupin, ampak imajo lahko lastnosti obeh skupin. Npr. števila, ki sodijo med kvantitativne podatke, so lahko sestavina kvalitativno opisanih podatkov. Kot primer navedimo točko v ravnini, ki je predstavljena z urejenim parom dveh števil, katerih zaloga vrednosti je $\mathbb{R}$. (Kvalitativni podatek vsebuje dva kvantitativna podatka.)

Razlikovanje med opisanima vrstama podatkov bo v nadaljevanju pomembno za modeliranje nedoločenosti oz. natančneje, za način medsebojnega razlikovanja situacij, kjer vsaka od njih predstavlja neko realizacijo podatkov. Takšnim situacijam bomo v prihodnje rekli tudi scenariji in razlikam med njimi perturbacije.

2.1.2 Diskretnost in zveznost

Diskretnost ali *zveznost* podatka nam pove, na koliko različnih načinov se lahko nek podatek pojavi v problemu. Najprej se osredotočimo le na kvantitativno opisane podatke.

Diskretnost podatkov. Gre za številske podatke, katerih zaloga vrednosti je diskretna množica, t.j. množica, za katero obstaja bijektivna preslikava z množico naravnih števil. Diskretni podatek lahko zavzame največ štev no mnogo različnih vrednosti.

Zveznost podatkov. Gre za podatke, katerih zaloga vrednosti je zvezna množica, t.j množica za katero obstaja bijektivna preslikava z množico realnih števil. Zvezni podatek lahko zavzame neštevno mnogo različnih vrednosti.

Kvantitativne podatke torej enostavno razvrstimo v eno izmed obeh skupin. V katero skupino pa sodijo kvalitativni podatki? Ti pravzaprav sodijo v skupino diskretnih podatkov. Npr. obstaja številno mnogo polnih dvodelnih grafov z $2n$ vozlišči itd. Težava se pojavi le pri sestavljenih podatkih, katerih sestavni deli niso vsi strogo v eni od skupin. V takem primeru pa pravimo, da je podatek diskreten, če so vsi njegovi sestavni deli diskretni; če je le en sestavni del podatka zvezen, potem pravimo, da je podatek zvezen. Npr. povezava grafa, katere dolžina je lahko poljubno realno število, je zvezen podatek.

Diskretnost ali zveznost podatkov, ki nastopajo v optimizacijskem problemu, je pogosto pomembna za izbiro metode oz. načina reševanja problema. Npr. tako imenovani zvezni problemi – ki vsebujejo zvezno opisane podatke – se večkrat rešujejo z metodami računske geometrije[1], lokalne optimizacije, simuliranega ohlajanja, matematičnega programiranja, nekonveksne optimizacije itd. Po drugi strani pa za reševanje diskretnih problemov – ki vsebujejo samo diskretno opisane podatke – večkrat uporabljamo metode, ki so osnovane na poglobljenem razumevanju strukture in odnosov med podatki. V nadaljevanju se bomo ukvarjali predvsem z diskretno opisanimi podatki.

2.1.3 Stalnost in spremenljivost

Delitev na skupini *stalnih* ali *statičnih* podatkov in *spremenljivih* ali *dinamičnih* podatkov izhaja iz časovne odvisnosti podatkov. Opišimo obe skupini.

Stalni ali statični podatki so podatki, ki se ne spreminjajo s časom. Pravimo, da so statični podatki časovno neodvisni; njihova vrednost je torej stalna in nespremenljiva. Pogosto definicijo razširimo tudi na problem in pravimo, da gre za statični problem, če so vsi vhodni podatki problema statični. Najpogosteje se statičnosti problema ne poudarja posebej.

Spremenljivi ali dinamični podatki so podatki, ki se spreminjajo s časom. Pravimo tudi, da so dinamični podatki odvisni od časa. Za problem pri katerem je vsaj en vhodni parameter odvisen od časa, pravimo, da je dinamični problem. Kadar raziskovalci obravnavajo dinamične probleme, to, v nasprotju s statičnimi problemi, posebej poudarijo.

Iz nekega statičnega optimizacijskega problema lahko izpeljemo njegovo dinamično različico preprosto tako, da v enega izmed vhodnih parametrov problema vpeljemo časovno odvisnost. Slednji so zaradi vsebovane časovne odvisnosti večkrat precej zahtevnejši za obravnavo. Večina v literaturi obravnavanih problemov sodi v skupino statičnih problemov.

[1] *angl. computational geometry*

2.1.4 Določenost in nedoločenost

Opišimo še zadnjo razdelitev na *določene* in *nedoločene* podatke. Nedoločenosti podatkov bomo pozornost namenili v zadnjem razdelku tega poglavja, na tem mestu si oglejmo le kratek opis razdelitve.

Določenost podatkov. Gre za podatke, katerih vrednost je točno določena. Takšna vrednost je popolnoma gotova, zanesljiva in se ne spreminja s časom.

Nedoločenost podatkov. Podatek, ki ni določen, je nedoločen. Gre za podatek, ki je podan npr. z neko verjetnostno porazdelitvijo ali z naborom vrednosti, ki jih lahko zavzame. V dinamičnih problemih je čas navadno nedoločen podatek, zato so dinamični podatki pravzaprav tudi nedoločeni.

Večina v literaturi obravnavanih problemov se ukvarja z optimizacijskimi problemi, v katerih nastopajo določeni podatki. Razlog za to gre iskati v zahtevnosti obravnave nedoločenosti. V nadaljevanju se bomo z nedoločenostjo še večkrat srečali.

2.1.5 Grafi in omrežja.

V tem podrazdelku bomo na kratko predstavili pomembnejše matematične strukture, predvsem grafe in omrežja ter z njima povezane pojme, ki jih pogosto uporabljamo v tem delu. Osnovno izrazje in pojme črpamo predvsem iz [WW97].

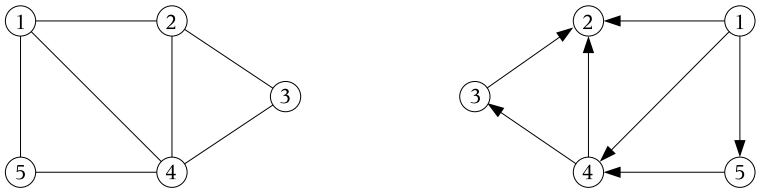
Neusmerjeni graf. *Neusmerjeni graf* G je urejen par (V, E). Označimo ga z $G = (V, E)$, kjer je V množica vozlišč in $E \subseteq V \times V$ množica dvosmernih povezav. Dvosmerna povezava $(u, v) \in E$ je neurejen par dveh vozlišč $u, v \in V$. V nadaljevanju bomo večkrat uporabljali tudi oznaki $n = |V|$ in $m = |E|$.

Digraf. *Usmerjeni graf* ali *digraf* je graf, v katerem so povezave usmerjene. Digraf pogosto označimo z $G = (V, A)$, kjer je V množica vozlišč in $A \subseteq V \times V$ množica usmerjenih povezav. Usmerjena povezava $(u, v) \in E$ je urejen par dveh vozlišč $u, v \in V$, pri čemer je vozlišče u začetek ali izvor povezave (u, v) in vozlišče v konec ali ponor povezave (u, v). V nadaljevanju bomo, kadar pri tem ne bo prihajalo do dvoumnosti, namesto usmerjeni ali neusmerjeni graf največkrat rekli kar graf.

Predstavitev grafov. Grafično neusmerjene in usmerjene grafe prikažemo tako kot na sliki 2.1. Oba grafa na sliki 2.1 sta definirana kot $G = (V, E)$, kjer je $V = \{1, 2, 3, 4, 5\}$ in $E = \{(1, 2), (1, 4), (1, 5), (3, 2), (4, 2), (4, 3), (5, 4)\}$, pri čemer gre v primeru a) za neusmerjeni graf in v primeru b) za digraf.

Graf lahko predstavimo tudi z *matriko sosednosti*, v kateri so vrstice in stolpci označeni z vozlišči. V i-ti vrstici in j-tem stolpcu matrike sosednosti nastopa 1, če

Slika 2.1: Neusmerjeni graf a) in digraf b)



a) b)

v G obstaja povezava $(i, j) \in E$, sicer pa 0. Če je graf enostaven, t.j. ne vsebuje *zank*, potem so diagonalni elementi matrike sosednosti enaki 0. Matrika sosednosti je simetrična za neusmerjene grafe. Na sliki 2.2 sta prikazani matriki sosednosti za grafa s slike 2.1.

Slika 2.2: Matriki sosednosti

$$\begin{pmatrix} 0 & 1 & 0 & 1 & 1 \\ 1 & 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 1 & 0 \\ 1 & 1 & 1 & 0 & 1 \\ 1 & 0 & 0 & 1 & 0 \end{pmatrix} \qquad \begin{pmatrix} 0 & 1 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \end{pmatrix}$$

a) b)

Omrežje. Omrežje je graf $G = (V, E)$, v katerem je vsaki povezavi $e \in E$ prirejena utež $w(e) \in \mathbb{R}$. Včasih omrežju zato rečemo tudi uteženi graf. Utež lahko predstavlja npr. dolžino povezave ali čas potovanja med dvema vozliščema. Glede na graf G razlikujemo usmerjena in neusmerjena omrežja. Matrika sosednosti za omrežje G v vrstici i in stolpcu j vsebuje $w((i, j))$, če $(i, j) \in E$, sicer pa ∞.

Sprehod in pot. *Sprehod* v grafu je zaporedje povezav grafa (digrafa), tako da je končno vozlišče vsake povezave v sprehodu enako začetnemu vozlišču naslednje povezave. Za sprehod $s = (e_1, e_2, \ldots, e_n)$, kjer $e_i \in E$, velja $e_i = (v_i, v_{i+1})$. *Pot* je sprehod, kjer vsaka povezava nastopa največ enkrat. Če za sprehod velja, da se končno vozlišče zadnje povezave v sprehodu ujema z začetnim vozliščem prve povezave, potem takemu sprehodu pravimo *cikel*.

V omrežjih lahko izračunamo tudi *dolžino sprehoda*, ki je definirana kot vsota dolžin povezav na tem sprehodu, t.j.

$$d(s) = \sum_{e_i \in s} w(e_i).$$

Večkrat nas za dve vozlišči $u, v \in V$ zanima dolžina najkrajše poti med u in v, ki jo označimo z $d(u, v)$. *Matrika dolžin najkrajših poti* je matrika, v kateri v i-ti vrstici in j-tem stolpcu nastopa dolžina najkrajše poti med vozliščema i in j.

Algoritmi za računanje matrike dolžin najkrajših poti v grafu so dobro znani [Meh84, CLRS01, Vil02]. Enostavno implementacijo omogoča t.i. množenje matrik po postopku Floyd-Warshalla, ki je predstavljeno z algoritmom 2.1 spodaj.

Algoritem 2.1: Floyd-Warshallovo množenje matrik

Vhod: matrika sosednosti M za omrežje $G = (V, E)$.
Izhod: matrika dolžin najkrajših poti D za G.

```
1. D := M;
2. for k := 1 to n do
3.     for i := 1 to n do
4.         for j := 1 to n do
5.             D_ij := min(D_ij, D_ik + D_kj);
6. return D;
```

Stopnja vozlišča. Število povezav, v katerih je eno od krajišč vozlišče $i \in V$, imenujemo *stopnja* vozlišča in jo označimo z $\delta(i)$. Za vozlišča digrafov posebej definiramo *vhodno* in *izhodno* stopnjo vozlišča, ki predstavlja število povezav, ki vstopajo oz. izstopajo iz vozlišča. Stopnja vozlišča digrafa je enaka seštevku vhodne in izhodne stopnje. *Izolirano* vozlišče ima stopnjo enako 0, stopnja *neizoliranega* vozlišča pa je vsaj 1.

Soseščina vozlišča in induciran graf. Za graf $G = (V, E)$ definiramo *odprto soseščino* vozlišča $u \in V$ kot

$$\mathcal{N}(u) = \{v \in V : (u, v) \in E\}.$$

$\mathcal{N}(u)$ torej vsebuje vsa vozlišča $v \in V$, ki so v G neposredno povezana z u. Včasih v soseščino $\mathcal{N}(u)$ dodamo še vozlišče u; v tem primeru takšni soseščini pravimo *zaprta soseščina* in jo označimo z

$$\mathcal{N}[u] = \mathcal{N}(u) \cup \{u\}.$$

Potrebovali bomo tudi pojem *induciranega grafa*. Za induciran graf $H(U) = (U, F)$ na grafu $G = (V, E)$ velja, da $U \subseteq V$ in $F = \{(u, v) \in E | u, v \in U\}$. Graf $H(U)$ je torej *podgraf* grafa G, kar označimo s $H(U) \subseteq G$.

2.2 Modeliranje problemov

2.2.1 Definicija optimizacijskega problema

Osnovne definicije in pojme s področja optimizacijskih problemov povzemamo po [ACG+99, Rob02]. Optimizacijski problem je vrsta računskega problema. V praksi se seveda najprej srečamo z nekim konkretnim primerom problema, kjer so namesto splošnih parametrov dejanski podatki. Tak konkretni problem imenujemo *naloga*.

Optimizacijski problem P je četverka $P = (I, S, m, cilj)$, kjer je

- I množica nalog problema P;
- S funkcija, ki vsaki nalogi $x \in I$ priredi množico $S(x)$ dopustnih rešitev;
- m kriterijska funkcija, ki vsaki dopustni rešitvi $y \in S(x)$ priredi vrednost $m(x, y) \in \mathbb{R}$;
- $cilj \in \{\min, \max\}$, ki pove, ali gre za minimizacijski ali maksimizacijski optimizacijski problem.

Naj bo $x \in I$ naloga problema $P = (I, S, m, cilj)$. Optimalne rešitve naloge x so dopustne rešitve $y^* \in S(x)$, za katere je $m(x, y^*) = cilj\{w|w = m(x, z) \wedge z \in S(x)\}$. Množico optimalnih rešitev naloge x označimo z $S^*(x)$, njihovo optimalno vrednost pa z $m^*(x) = m(x, y^*)$.

Optimizacijski problem $P = (I, S, m, cilj)$ se lahko pojavi v eni od treh oblik, ki se razlikujejo po tem, kaj je potrebno izračunati:

- *konstrukcijska* oblika P_{kon} zahteva, da za dano nalogo x izračunamo $y^*(x)$ in vrednost $m^*(x)$;
- *nekonstrukcijska* oblika P_{nek} zahteva, da za dano nalogo x izračunamo $m^*(x)$;
- *odločitvena* oblika P_{odl} zahteva, da za dano nalogo x in konstanto K ugotovimo, ali je optimalna vrednost "boljša" od K, t.j. ali je $m^*(x) \leq K$, če $cilj = \min$, oz. $m^*(x) \geq K$, če $cilj = \max$.

Pri odločitveni obliki optimizacijskega problema je množica možnih rešitev zelo enostavna, $S(x) = \{DA, NE\}$, kar je razlog, da je bila ta oblika najprej vzeta pod drobnogled. Več o odločitvenih oblikah se nahaja v [GJ79, HMU01]. V nadaljevanju se bomo največ ukvarjali s konstrukcijskimi in odločitvenimi oblikami optimizacijskih problemov. Slednje bomo uporabljali predvsem za dokazovanje $\mathcal{NP}$-težkosti prvih.

Vse optimizacijske probleme, obravnavane v tem delu, bomo definirali na enak način kot v [ACG+99]. Najprej bomo opisali nalogo, nato obliko dopustnih rešitev, kriterijsko funkcijo in še cilj, ki je lahko minimizacija ali maksimizacija.

2.2.2 Zahtevnost algoritmov

Računski modeli

Teorija zahtevnosti algoritmov se ukvarja predvsem z analizo časa in prostora, ki ga nek algoritem potrebuje za izvedbo. Ker sta čas in prostor v praksi močno odvisna od strukturnih in tehnoloških značilnosti strojne opreme, nas običajno ne zanimajo lastnosti algoritma na konkretnem računalniku, ampak na nekem abstraktnem računalniškem *modelu*. Pri analizi algoritma se raje opiramo na ta model, ob tem pa se seveda zavedamo, da se bo izvajanje algoritma na konkretnem računalniku za nek konstantni faktor razlikovalo od izvajanja na modelu.

Razlikujemo dve vrsti modelov, t.j. *modele z logaritemskim cenikom*, kjer se vsaka operacija izvede v času, ki je odvisen od velikosti operandov (navadno sorazmerno s številom bitov v vseh operandih), in *modele z enakomernim cenikom*, kjer se vsaka operacija izvede v konstantnem času, ne glede na velikost operandov. Računski čas algoritma je torej sorazmeren s številom izvedenih operacij. Podobne modele kot za merjenje časa uporabljamo tudi za merjenje prostora.

Logaritemski cenik je primernejši, kadar števila, ki nastopajo v algoritmu naraščajo zelo hitro. V nasprotnem primeru pa je povsem na mestu tudi enakomerni cenik, saj se ni bati, da bi postal model s takim cenikom nesmiselno močan. V nadaljevanju bomo uporabljali izključno enakomerni cenik.

Velikost naloge

Če želimo problem rešiti z računskim strojem, moramo nalogo zapisati na stroju razumljiv način. Nalogo $x \in I$ zapišemo (zakodiramo) z nizom $e(x)$, ki ga sestavljajo simboli strojeve vhodne abecede Σ. Dolžina niza $e(x)$ je mera za *velikost naloge* x. Dolžino niza $e(x)$ bomo označili z $|e(x)|$, velikost naloge x pa z $|x|$. Velja torej $|x| = |e(x)|$.

Funkcijo $e : I \to \Sigma^*$ imenujemo *kodirna shema*. Med vsemi kodirnimi shemami nas bodo zanimale predvsem *naravne* kodirne sheme, t.j. takšne, ki zadoščajo naslednjim zahtevam:

- kodiranje ne vključuje reševanja;
- kodiranje ne uporablja eniškega zapisa števil, temveč mestni številski sistem z osnovo $b \geq 2$;
- kodiranje tudi na kak drug način ne napihuje niza;
- kodiranje ne otežuje dekodiranja in preverjanja rezultatov.

Asimptotična analiza algoritma

V praksi nas ne zanima natančen računski čas (oz. prostor) algoritma za vsako posamično nalogo, temveč kako hitro narašča računski čas, če narašča velikost nalog.

Vendar imajo nekateri algoritmi pri enako velikih nalogah zelo različne računske čase. Zato uporabljamo predvsem naslednja dva pristopa:

- *analiza za najslabši primer*, ki ugotavlja, kako hitro narašča najslabši računski čas – gre za pesimistični pristop;
- *analiza za povprečni primer*, ki ugotavlja, kako hitro narašča povprečni oz. pričakovani računski čas – pri tej analizi je potrebno upoštevati verjetnostno porazdelitev nalog, zato je ta analiza pogosto zelo zahtevna.

Pri analizi časovne ali prostorske zahtevnosti algoritma nas navadno zanima njegova *asimptotična zahtevnost*, t.j. zahtevnost v odvisnosti od naraščanja velikosti naloge. Za takšne izjave uporabljamo O-zapis. Naj bosta dani funkciji $f, g : \mathbb{N} \to \mathbb{N}$. Pravimo, da je

- f *kvečjemu reda* g, oziroma $f(n) = \emptyset(g(n))$, če obstajata konstanti c in n_0, da je $|f(n)| \leq c|g(n)|$ za vsak $n > n_0$;
- f *vsaj reda* g, oziroma $f(n) = \Omega(g(n))$, če obstajata konstanti c in n_0, da je $|f(n)| \geq c|g(n)|$ za vsak $n > n_0$;
- f *reda* g, oziroma $f(n) = \Theta(g(n))$, če je $f(n) = \emptyset(g(n))$ in $f(n) = \Omega(g(n))$.

Oglejmo si časovno zahtevnost algoritma na primeru Floyd-Warshallovega algoritma 2.1. Njegova (povprečna in najslabša) časovna zahtevnost je $\Theta(n^3)$, ker vsebuje tri ugnezdene zanke, kjer vsaka zahteva n korakov za izvedbo. Njegova prostorska zahtevnost pa je $\Theta(n^2)$, ker algoritem za izvajanje potrebuje matriko D, katere velikost je $n \times n$.

Nedeterministični algoritmi

Delovanje vsakega algoritma si lahko predstavljamo s pomočjo *računskih dreves*. Osredotočimo se na odločitvene probleme. Vsaki nalogi pripada neko računsko drevo, vsako vozlišče v drevesu pa predstavlja neko stanje, v katerem se nahaja algoritem (oz. stroj, ki ta algoritem izvaja) med reševanjem te naloge. Algoritem ima lahko v nekem stanju na izbiro več potez (povezav); izbor katerekoli od njih vodi v novo stanje. Listi računskega drevesa so stanja, od koder algoritem ne more več nadaljevati, bodisi zato, ker je izvedel odgovor (DA ali NE), ali pa ker zaradi nekega razloga nima na voljo nobene poteze več (takrat je smiselno predpostavljati odgovor NE). Če obstaja vsaj en list, v katerem je odgovor DA, potem končni odgovor algoritma na nalogo mora biti DA, sicer pa – kadar so v vseh listih odgovori NE – je edino smiselno, da algoritem odgovori NE.

Običajnemu, t.j. *determinističnemu* algoritmu v splošnem ne preostane nič drugega, kot da na nek sistematičen način pregleduje možne poti iz korena do listov, dokler bodisi ne najde lista z odgovorom DA (kadar ta obstaja) bodisi ne pregleda vseh listov (kadar je v vseh listih odgovor NE).

Možno pa je tudi drugače. *Nedeterminističen* algoritem ima "čudežno" moč, ki mu v vsakem od stanj omogoča, da ugane pravo potezo do lista z odgovorom DA. Če odgovora DA ni v nobenem listu, mora nedeterminističen algoritem še vedno odgovoriti NE. Med ugibanjem potez v stanjih računskega drevesa nedeterministični algoritem sestavlja še ustrezno konstruktivno rešitev $y(x)$.

Nedeterministični algoritem lahko simuliramo z deterističnim, ki namesto ugibanja sistematično pregleduje drevo, vendar je tak algoritem v splošnem počasnejši. Če nedeterministični algoritem reši nalogo v času $t(n)$, potem deterministični gotovo lahko reši nalogo v času $\emptyset(2^{t(n)})$. To pa je lahko bistvena upočasnitev in v primeru, ko je $t(n)$ polinom, lahko pomeni razliko med praktično obvladljivostjo in neobvladljivostjo problema.

2.2.3 Zahtevnost problemov

Definicija

V nadaljevanju nas bo večkrat zanimala asimptotična zahtevnost problemov, ki jo definiramo s pomočjo asimptotične zahtevnosti algoritmov.

- $\emptyset(f(n))$ je *zgornja meja* časovne (prostorske) zahtevnosti problema P, če za P *obstaja* algoritem s časovno (prostorsko) zahtevnostjo $\emptyset(f(n))$;
- $\Omega(f(n))$ je *spodnja meja* časovne (prostorske) zahtevnosti problema P, če za P *vsak* algoritem zahteva vsaj $\Omega(f(n))$ časa (prostora).

S pomočjo časovne (prostorske) zahtevnosti odločitvenih problemov lahko definiramo različne razrede odločitvenih problemov. (Razrede bomo kasneje razširili tudi na optimizacijske probleme.) $\mathcal{DTIME}(f(n))$ je razred odločitvenih problemov, ki so rešljivi v času $\emptyset(f(n))$; podobno je $\mathcal{DSPACE}(f(n))$ razred odločitvenih problemov, ki so rešljivi na prostoru velikosti $\emptyset(f(n))$; in nadalje je $\mathcal{NTIME}(f(n))$ razred odločitvenih problemov, ki so nedeterministično rešljivi v času $\emptyset(f(n))$.

Razredi problemov

S pomočjo teh definicij nadalje izpeljemo širše razrede:

- $\mathcal{P} = \cup_{k\geq 0}\mathcal{DTIME}(n^k)$ je razred odločitvenih problemov, deterministično rešljivih v polinomskem času glede na velikost naloge;
- $\mathcal{PSPACE} = \cup_{k\geq 0}\mathcal{DSPACE}(n^k)$ je razred odločitvenih problemov, deterministično rešljivih na polinomsko velikem prostoru glede na velikost naloge;
- $\mathcal{EXPTIME} = \cup_{k\geq 0}\mathcal{DTIME}(2^{n^k})$ je razred odločitvenih problemov, deterministično rešljivih v eksponentnem času glede na velikost naloge;
- $\mathcal{NP} = \cup_{k\geq 0}\mathcal{NTIME}(n^k)$ je razred odločitvenih problemov, nedeterministično rešljivih v polinomskem času glede na velikost naloge.

Velja naslednje

$$\mathcal{P} \subseteq \mathcal{NP} \subseteq \mathcal{PSPACE} \subseteq \mathcal{EXPTIME}. \tag{2.1}$$

Ker velja $\mathcal{P} \subsetneq \mathcal{EXPTIME}$, velja, da je vsaj ena od relacij v (2.1) prava podmnožica, a še ni znano katera.

Vpeljimo še dva razreda optimizacijskih problemov. Optimizacijski problem $P = (I, S, m, cilj)$ je v razredu $\mathcal{NPO}$, če velja:

- množica nalog I je razpoznavna v polinomskem času, t.j. za vsak x lahko v polinomskem času glede na $|x|$ ugotovimo, ali $x \in I$;
- dopustne rešitve so polinomsko dolge in razpoznavne v polinomskem času, t.j. obstaja polinom q, da za vsako nalogo $x \in I$ in vsako dopustno rešitev $y \in S(x)$ velja $|y| \leq q(|x|)$, ter da lahko za vsak z, kjer $|z| \leq q(|x|)$, v polinomskem času glede na $|z|$ ugotovimo, ali $z \in S(x)$;
- kriterijska funkcija m je izračunljiva v polinomskem času, t.j. če $x \in I$ in $y \in S(x)$, potem lahko $m(x, y)$ izračunamo v polinomskem času glede na $|x|$ in $|y|$.

Optimizacijski problem $P = (I, S, m, cilj)$ je v razredu $\mathcal{PO}$, če velja:

- $P \in \mathcal{NPO}$;
- P_{kon} je rešljiva v polinomskem času.

Optimizacijskim problemom v razredu $\mathcal{NPO}$ ustrezajo odločitveni problemi v $\mathcal{NP}$; podobno optimizacijskim problemom v $\mathcal{PO}$ ustrezajo odločitveni problemi v $\mathcal{P}$. Velja naslednja trditev, katere dokaz se nahaja v [Rob02].

Trditev 2.1 ▶ $P \in \mathcal{NPO} \Longrightarrow P_{odl} \in \mathcal{NP}$.

Karpove in Turingove prevedbe

Naj bosta dana dva odločitvena problema P_1 in P_2. Pravimo, da problem P_1 *prevedemo* na P_2, če znamo vsako nalogo problema P_1 rešiti na posreden način preko problema P_2. Natančneje, odločitveni problem P_1 je *prevedljiv po Karpu*[2] na odločitveni problem P_2, kar zapišemo $P_1 \preceq_m P_2$, če obstaja algoritem R, ki vsako nalogo x problema P_1 pretvori v tako nalogo $R(x)$ problema P_2, da je x pozitivna natanko takrat, kadar je $R(x)$ pozitivna. Takšen R se imenuje *Karpova prevedba* P_1 na P_2. Posebej pomembna situacija je, ko ima R polinomsko časovno zahtevnost. V tem primeru pravimo, da je problem P_1 *polinomsko prevedljiv po Karpu* na P_2.

[2] V angl. tudi many-to-one.

Karpova prevedba poveže le pozitivne (oz. negativne) naloge enega odločitvenega problema s pozitivnimi (oz. negativnimi) nalogami drugega. Za prevajanje med optimizacijskimi problemi pa je potrebna splošnejša, t.i. *Turingova prevedba*. Turingova prevedba predpostavlja samo to, da problem P_1 znamo rešiti že, če znamo rešiti problem P_2, pri čemer ni pomembno, kako zahtevno in kako pogosto je vmesno reševanje problema P_2.

Definirajmo najprej *preroka* za problem P_2, ki je abstraktna naprava, ki za vsako nalogo problema P_2 v enem koraku vrne njeno rešitev. Problem P_1 je *prevedljiv po Turingu* na problem P_2, kar zapišemo $P_1 \preceq_T P_2$, če obstaja algoritem R, ki reši P_1 s pomočjo preroka za P_2. Takšen R imenujemo *Turingova prevedba* P_1 na P_2. Posebno nas bodo zanimale situacije, ko ima R polinomsko časovno zahtevnost. V tem primeru pravimo, da je problem P_1 *polinomsko prevedljiv po Turingu* na P_2, kar zapišemo $P_1 \preceq_T^p P_2$.

$\mathcal{NP}$-polnost in $\mathcal{NP}$-težkost

Razred $\mathcal{C}$ je *zaprt* za dano prevedbo $\preceq$, če velja, da kadar je nek problem v $\mathcal{C}$, je z njim v $\mathcal{C}$ tudi vsak nanj $\preceq$-prevedljiv problem. Natančneje, razred $\mathcal{C}$ je zaprt za prevedbo $\preceq$, če za vsak par problemov P_1 in P_2 velja $P_1 \preceq P_2 \wedge P_2 \in \mathcal{C} \Rightarrow P_1 \in \mathcal{C}$.

Problem P_2 iz razreda $\mathcal{C}$ je *poln* v $\mathcal{C}$ glede na prevedbo $\preceq$, če za vsak drug problem P_1 iz $\mathcal{C}$ velja $P_1 \preceq P_2$. Problem P_2 tedaj imenujemo tudi $\mathcal{C}$-poln glede na $\preceq$. Velja naslednja trditev.

Trditev 2.2 ▶ *Naj bo $\mathcal{C}_1 \subseteq \mathcal{C}_2$ in naj bo razred $\mathcal{C}_1$ zaprt za prevedbo $\preceq$. Potem je $\mathcal{C}_1 \subsetneq \mathcal{C}_2$ natanko tedaj, ko $\mathcal{C}_2$-polni problemi (glede na $\preceq$) niso v $\mathcal{C}_1$.*

Dokaz trditve lahko zopet najdemo v [Rob02]. Zanimiva je uporaba trditve na razredih $\mathcal{P}$ in $\mathcal{NP}$. Vemo že, da velja bodisi $\mathcal{P} \subsetneq \mathcal{NP}$ ali $\mathcal{P} = \mathcal{NP}$. Z uporabo zgornje trditve lahko zapišemo

$$\mathcal{P} = \mathcal{NP} \iff \text{obstaja } \mathcal{NP}\text{-poln problem, ki je v } \mathcal{P}.$$

Velja seveda tudi obratna trditev.

$\mathcal{NP}$-polni odločitveni problemi so torej najtežji problemi v razredu $\mathcal{NP}$. V praksi se prav radi pojavijo in tudi večina v nadaljevanju obravnavanih problemov sodi v to skupino. Kljub njihovi številnosti in raznovrstnosti vprašanje enakosti oz. različnosti $\mathcal{P}$ in $\mathcal{NP}$, še vedno ostaja odprto, vendar se čedalje bolj utrjuje prepričanje, da je $\mathcal{P} \neq \mathcal{NP}$.

Problemi v razredu $\mathcal{P}$ so "lahko" obvladljivi, kar pomeni da za njih obstajajo algoritmi, ki v polinomskem času vrnejo optimalno rešitev. Če pa je problem v $\mathcal{NP}$ in ni v $\mathcal{P}$, ali še posebej, če je problem $\mathcal{NP}$-poln, potem je zelo malo verjetno, da bi našli polinomski algoritem, ki bi vračal optimalne rešitve. To je razlog, da $\mathcal{NP}$-polne probleme smatramo za "težko" obvladljive.

Prvi problem, za katerega je bila dokazana $\mathcal{NP}$-polnost, je izpolnljivost Boolovih izrazov [Coo71]. Neposredno dokazovanje $\mathcal{NP}$-polnosti je precej zahtevno opravilo, zato jo večkrat raje skušamo dokazati na posreden način. $\mathcal{NP}$-polnost nekega odločitvenega problema P_2 lahko dokažemo v dveh korakih:

1. pokažemo, da je $P_2 \in \mathcal{NP}$; in

2. na P_2 prevedemo po Karpu nek znan $\mathcal{NP}$-poln problem P_1.

Kako pa je z zahtevnostjo reševanja optimizacijskih problemov? Optimizacijski problem P je $\mathcal{NP}$-težek, če za vsak odločitveni problem $P' \in NP$ velja $P' \preceq_T^p P$. Velja naslednja trditev.

Trditev 2.3 ▶ *Če $P \in \mathcal{NPO}$ in P_{odl} je $\mathcal{NP}$-poln $\Rightarrow$ P je $\mathcal{NP}$-težek.*

Dokaz. P_{odl} je prevedljiv na P, ker rešitev problema P_{odl} lahko enostavno ugotovimo iz rešitve problema P. Na P_{odl} pa so prevedljivi vsi problemi $P' \in \mathcal{NP}$, ker je sam P_{odl} $\mathcal{NP}$-poln. □

Torej če hočemo dokazati $\mathcal{NP}$-težkost optimizacijskega problema P, je dovolj, da pokažemo $\mathcal{NP}$-polnost odločitvenega problema P_{odl}. To dejstvo bomo veliko uporabljali v nadaljevanju. Obsežen spisek odločitvenih problemov, za katere je že dokazana $\mathcal{NP}$-polnost se nahaja v [GJ79]. Podoben spisek[3] $\mathcal{NP}$-težkih optimizacijskih problemov pa v [ACG+99].

2.3 Reševanje problemov

2.3.1 Analiza problemov

Recimo, da smo soočeni z nekim optimizacijskim problemom P, naša naloga pa je ugotoviti, kakšne so praktične možnosti za njegovo učinkovito reševanje. Najboljša možnost je, da se izkaže $P \in \mathcal{P}$, kar pravzaprav pomeni, da za P poznamo polinomski natančni algoritem. V tem primeru P sodi med "lahko" obvladljive probleme.

Druga možnost je, da za P ugotovimo, da gre za $\mathcal{NP}$-težek problem. Potemtakem je skoraj neverjetno (verjamemo, da je celo nemogoče), da bi zanj odkrili polinomski algoritem, ki bi vračal optimalne rešitve. Včasih rečemo tudi, da P sodi med "težko" obvladljive probleme. Kaj pa lahko storimo v tem primeru? Reševanja P se v grobem lahko lotimo na dva načina. Prvi način je, da za P skušamo poiskati superpolinomski[4] natančni algoritem in pri tem upamo, da njegova dejanska časovna zahtevnost na konkretnih primerih problema kljub vsemu ne bo prevelika. Drugi način je, da za P skušamo poiskati algoritem, ki v polinomskem času vrača

[3] Spisek je dostopen tudi na http://www.nada.kth.se/~viggo/problemlist/.

[4] Algoritem, katerega časovna zahtevnost je večja od polinomske.

približne rešitve. Več o približnem reševanju bomo povedali v naslednjem podrazdelku.

Preostane še tretja možnost, da za P ne najdemo niti polinomskega natančnega algoritma niti ne znamo pokazati njegove $\mathcal{NP}$-težkosti. Taki primeri se v praksi zelo redko pojavijo in tudi vsi v nadaljevanju obravnavani problemi sodijo v eno izmed prvih dveh skupin. Radovednega bralca pa za raziskovanje zadnje možnosti usmerjamo na [ACG+99, GJ79, HMU01, KT06].

2.3.2 Približno reševanje

Približnemu reševanju večkrat pravimo tudi *hevristično reševanje*, algoritmom za približno reševanje pa *hevristični algoritmi* ali kar *hevristike*. Hevristični algoritem za dani problem P vrača približne rešitve in le v izjemnih primerih (če imamo srečo) vrne optimalno. Približna rešitev problema je pravzaprav poljubna dopustna rešitev. V splošnem hevristično algoritem ne daje nobenih zagotovil za kakovost vrnjenih rešitev, kar pomeni, da je približna rešitev lahko poljubno slaba v primerjavi z optimalno.

Hevristične algoritme razvrstimo v naslednje skupine.

Aproksimacijski algoritmi. Sem sodijo algoritmi, ki za kakovost vrnjene rešitve dajejo tudi neko zagotovilo, ki se navadno izraža v primerjavi z optimalno rešitvijo. V nadaljevanju se bomo s to vrsto algoritmov veliko ukvarjali, zato jim bomo pozornost posebej namenili v naslednjem podrazdelku.

Verjetnostni algoritmi. Gre za algoritme, ki temeljijo na naključnosti. Za analizo algoritma je potrebno poznati verjetnostno porazdelitev vhoda. Poznamo *Las Vegas* algoritme, ki vedno odgovorijo pravilno, le verjetnost za njihovo dolgotrajno izvajanje je majhna, in *Monte-Carlo* algoritme, ki se vedno hitro izvedejo, vendar dopuščajo majhno verjetnost napačnega odgovora. Več o verjetnostnih algoritmih je v [ACG+99].

Metahevristični algoritmi. Gre za splošne algoritmične metode, ki jih lahko z malo prilagoditvami uporabimo za dani optimizacijski problem. Gre za metode, kot so: simulirano ohlajanje, iskanje s tabuji, lokalno iskanje, genetski algoritmi, evolucijsko programiranje, umetne nevronske mreže itd. Več o metahevristikah je v [Ree95].

2.3.3 Aproksimacijski algoritmi

Algoritem A je *aproksimacijski algoritem* za problem $P = (I, S, m, cilj)$, če za vsako nalogo $x \in I$ vrne neko dopustno rešitev $A(x) \in S(x)$. V praksi smo za dobro približno rešitev pripravljeni priznati le tako dopustno rešitev $A(x)$, katere vrednost

$m(x, A(x))$ se le malo razlikuje od optimalne vrednosti $m^*(x)$. Razliko med vrednostjo $m(x, A(x))$ približne rešitve in vrednostjo $m^*(x)$ lahko ocenjujemo na več načinov. Naštejmo jih.

- *Absolutna napaka* rešitve y naloge x je

$$D(x, y) = |m^*(x) - m(x, y)|.$$

- *Relativna napaka* rešitve y naloge x je

$$E(x, y) = \frac{D(x, y)}{\max\{m^*(x), m(x, y)\}}.$$

  Tako za minimizacijske kot maksimizacijske probleme velja $0 \leq E(x, y) \leq 1$.

- *Performančni kvocient* rešitve y naloge x je

$$R(x, y) = \max\left\{\frac{m(x, y)}{m^*(x)}, \frac{m^*(x)}{m(x, y)}\right\}.$$

  Velja $1 \leq R(x, y)$.

Meri $E(x, y)$ in $R(x, y)$ sta povezani z naslednjo relacijo

$$E(x, y) = 1 - 1/R(x, y).$$

V nadaljevanju bomo uporabljali le meri $D(x, y)$ in $R(x, y)$.

Definirajmo še dve posebni zvrsti aproksimacijskih algoritmov. Algoritem A je *absolutni aproksimacijski algoritem* za problem $P = (I, S, m, cilj)$, če obstaja konstanta $d \geq 0$, da je $D(x, A(x)) \leq d$ za vsako nalogo $x \in I$. Algoritem A je *ρ-aproksimacijski algoritem* za problem $P = (I, S, m, cilj)$, če obstaja konstanta $\rho \geq 1$, da je $R(x, A(x)) \leq \rho$ za vsako nalogo $x \in I$. Konstanti ρ pravimo tudi *aproksimacijski faktor* algoritma A.

Definirajmo še razred optimizacijskih problemov, za katere obstaja aproksimacijski algoritem s konstantnim aproksimacijskim faktorjem. Optimizacijski problem $P \in \mathcal{NPO}$ je ρ-aproksimabilen, če zanj obstaja ρ-aproksimacijski algoritem z $\rho < \infty$. Razred $\mathcal{APX}$ je razred ρ-aproksimabilnih problemov iz $\mathcal{NPO}$. Velja naslednja trditev.

Trditev 2.4 ▶ $\mathcal{P} \neq \mathcal{NP} \Rightarrow \mathcal{APX} \subsetneq \mathcal{NPO}$.

Z drugimi besedami, obstajajo optimizacijski problemi, za katere ne obstaja ρ-aproksimacijski algoritem. Primer takšnega problema je problem Trgovski Potnik [Rob02].

Pojem aproksimabilnosti lahko razširimo na poljubno funkcijo $\rho(n)$ in pravimo, da je problem $\rho(n)$-aproksimabilen, če zanj obstaja aproksimacijski algoritem, da velja $R(x, A(x)) \leq \rho(n)$ za vsako nalogo $x \in I$, kjer $n = |x|$.

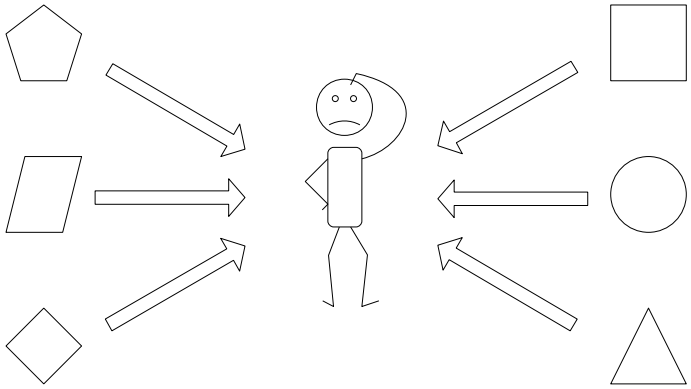
2.4 Nedoločenost v optimizacijskih problemih

2.4.1 Kaj je nedoločenost?

Posvetimo sedaj pozornost nedoločenosti podatkov v optimizacijskih problemih. Nek podatek je *nedoločen*, če ne moremo povedati, kakšna je njegova natančna vrednost. V nasprotnem primeru je seveda *določen*.

Z nedoločenostjo se večkrat srečamo npr. v ekonomiji, kjer se menedžerji odločajo o prihodnjih strategijah podjetja itd. Na primer oseba na sliki 2.3 sprejema pomembne odločitve v povezavi s prihodnjim razvojem izdelka, vendar nedoločenost vhodnega podatka, ki se lahko realizira kot krog, kvadrat ali kot eden izmed ostalih likov, prikazanih na sliki, osebi povzroča težave pri odločanju.

Slika 2.3: Nedoločenost vhodnih podatkov.



Kadar govorimo o nedoločenosti vhodnega podatka, je o takem podatku vedno poznanih vsaj nekaj informacij; pravzaprav nikoli ne gre za popolno nedoločenost podatka, ker o takem podatku ne moremo ničesar povedati. Matematično lahko nedoločenost podatka opišemo na več različnih načinov, npr. kot:

- nabor vrednosti, ki jih podatek lahko zavzame – na ta način lahko opišemo nedoločenost diskretnih podatkov;
- interval vrednosti, ki jih podatek lahko zavzame – omogoča opis nedoločenosti zveznih podatkov;
- diskretna ali zvezna slučajna spremenljivka – v tem primeru je potrebno poznavanje verjetnostne porazdelitve in s tem torej poznavanje stohastične narave problema;
- s funkcijo $f(t)$, kjer t lahko predstavlja čas v dinamičnih problemih – gre torej za nedoločenost zaradi dinamične narave problema;
- s scenariji, kjer vsak od scenarijev predstavlja možno realizacijo vhodnih podatkov – edino ta predstavitev izmed naštetih omogoča opis nedoločenosti kvalitativnih podatkov.

Kot vidimo, ima vsak izmed načinov predstavitve nedoločenosti svoje prednosti in slabosti, poleg tega so nekateri načini splošnejši in vsebujejo druge načine. Tako npr. opis nedoločenosti s funkcijo $f(t)$ zajema predstavitvi z naborom in intervalom vrednosti. S scenariji lahko predstavimo vse vrste nedoločenosti diskretnih podatkov. Scenariji omogočajo tudi predstavitev odvisnosti med posameznimi nedoločenimi podatki. Če so podatki med seboj neodvisni, je lahko predstavitev nedoločenosti s scenariji tudi zelo neučinkovita; npr. naj bosta $x \in \{x_1, x_2, \ldots, x_n\}$ in $y \in \{y_1, y_2, \ldots, y_m\}$ dva neodvisna podatka (predstavljena z naborom vrednosti), njuna predstavitev s scenariji bi zahtevala nm scenarijev. Kljub temu je predstavitev nedoločenosti s scenariji pogosto uporabljena, ker omogoča enostavno in zelo splošno modeliranje nedoločenosti, poleg tega edina omogoča predstavitev nedoločenosti kvalitativnih podatkov. To je tudi razlog, da bomo ta način v nadaljevanju uporabljali tudi mi.

2.4.2 Vzroki nedoločenosti

Večina v literaturi obravnavanih optimizacijskih problemov pravzaprav vsebuje določene podatke kljub dejstvu, da v praktičnih problemih velikokrat nastopajo nedoločeni podatki. Razlog za to tiči v dejstvu, da so problemi, ki vsebujejo nedoločenost, navadno bolj zahtevni za teoretično obravnavo. Kadar odstopanje med različnimi vrednostmi, ki jih podatek lahko zavzame, ni veliko, lahko nedoločenost podatka celo zanemarimo. Pogosto zanemarjanje nedoločenosti podatkov vodi v slabše modele realnih problemov in posledično do njihovih nekakovostnih rešitev.

Nedoločenost je lahko posledica samega zajema podatkov. Sem sodijo nenatančne in nezanesljive meritve, človeške napake itd. To vrsto nedoločenosti lahko zmanjšamo z uporabo bolj natančnih ali bolj zanesljivih merilnih naprav, z dvojnim preverjanjem itd. Vendar lahko cena dodatnega napora za izboljšanje meritve na ta način preseže razumno mejo. Če je temu tako, potem nam ne preostane drugega, kot nedoločenost sprejeti kot del problema.

Drugi vzrok nedoločenosti se večkrat skriva v sami stohastični naravi modeliranega problema, npr. modeliranje procesov, v katerih je prisoten človeški faktor, zapletenost ali nepoznavanje modeliranega procesa, uporaba novih ali netestiranih materialov in postopkov, nezmožnost napovedovanja prihodnosti, reakcija konkurence, spremembe modnih trendov in okusov strank, nihanje povpraševanja in ponudbe, nestabilnost obrestnih mer itd. V tem primeru pa je obravnava nedoločenosti kot dela problema edina rešitev.

Kot vidimo, je določenost vseh vhodnih podatkov v danem praktičnem problemu cilj, ki ga je pogosto težko, neekonomično ali celo nemogoče doseči. Zato je raziskovanje metod, ki omogočajo obvladovanje nedoločenosti vhodnih podatkov v optimizacijskih problemih, več kot smiselno.

2.4.3 Scenariji in perturbacije

Scenariji

Kot smo že omenili, bomo za predstavitev nedoločenosti vhodnih podatkov v optimizacijskih problemih uporabljali scenarije. *Scenarij* predstavlja neko realizacijo vhodnih podatkov danega problema. Nedoločenost podatkov je seveda opisana z vsaj dvema scenarijema. Naš cilj je poiskati kakovostno rešitev z upoštevanjem vseh možnih realizacij podatkov, torej vseh scenarijev.

Le več scenarijev skupaj lahko predstavlja nedoločenost podatkov; en sam scenarij pravzaprav predstavlja določene podatke. Z vidika enega samega scenarija gre torej za "osnovni" optimizacijski problem, z vidika več scenarijev pa za "nedoločeni" optimizacijski problem, ki je izpeljan iz osnovnega. Pri tem je potrebno posebno pozornost nameniti kriterijski funkciji izpeljanega optimizacijskega problema. Glede tega sta se v literaturi uveljavila naslednja dva pristopa:

- *robustna optimizacija* – najdena rešitev je "primerna" za vse scenarije; in
- *stohastična optimizacija* – najdena rešitev je bolj primerna za bolj verjetne in manj primerna za manj verjetne scenarije.

Oba pristopa bomo na kratko opisali v naslednjih dveh podrazdelkih. Kasneje bomo vpeljali še tretji način – *prilagodljivost*, v katerem bo za vsak scenarij potrebno poiskati svojo rešitev, pri čemer si bodo najdene rešitve kar najbolj podobne.

Pred tem si za ponazoritev oglejmo naslednji primer problema iz [SM96], v katerem nastopa nedoločenost podatkov. Gre za problem razmeščanja gasilskih postaj v Barceloni. Odzivni čas gasilcev je čas, potreben za intervencijo gasilske enote, ki je mestu požara najbližja. Ker je gostota mestnega prometa močno odvisna od naključnih dejavnikov, je odzivni čas gasilcev do poljubnega mesta nedoločen. Kljub temu vemo, da je promet gostejši ob prometnih konicah, t.j. v času prihodov in odhodov zaposlenih na delovna mesta. Tako lahko npr. definiramo dva scenarija, enega za prometne konice in drugega za normalno gostoto prometa. Cilj problema je razmestiti gasilske postaje tako, da je v obeh scenarijih želeni čas intervencije manjši od podanega.

Scenariji so uporabni za predstavitev nedoločenosti tako kvantitativnih kot kvalitativnih podatkov, npr. v enem scenariju neka utežena povezava v grafu nastopa, v drugem pa ne. Za modeliranje dinamičnih problemov, v katerih je čas diskretna spremenljivka, lahko prav tako uporabimo scenarije; pri tem vsak od scenarijev predstavlja nek časovni trenutek. V dinamičnih problemih gre navadno za neko linearno sosledje dogodkov. Modeliranje s scenariji pa nam pravzaprav omogoča celo poljubno sosledje dogodkov, ki ga ponazorimo z grafom, kjer vsakemu vozlišču pripišemo en scenarij. Več o tem v nadaljevanju.

Pri modeliranju nedoločenosti s scenariji je seveda pomembno, da nastopa le neko končno število scenarijev. Če scenarijem dodatno pripišemo še verjetnosti realizacije, na ta način pravzaprav vpeljemo diskretne slučajne spremenljivke. Nekateri

raziskovalci v primeru poznavanja verjetnosti realizacije, namesto o nedoločenosti raje govorijo o *tveganju* ali tudi o *stohastičnosti*.

V nadaljevanju bomo scenarije označevali z malo črko s in indeksom i, tako s_i predstavlja i-ti scenarij. Vhod "nedoločenega" problema je torej množica scenarijev $I = \{s_1, s_2, \ldots, s_n\}$. Z $S(s_i)$ bomo označevali dopustne rešitve za scenarij $s_i \in I$ z vidika "osnovnega" optimizacijskega problema. Pri tem bo $m(s_i, S)$ kakovost rešitve $S \in S(s_i)$ za scenarij s_i.

Perturbacije

V splošnem se lahko scenariji med seboj poljubno razlikujejo. Vendar se pogosto zgodi, da so si na nek način podobni. Podobnost scenarijev definiramo s pomočjo *perturbacij*. Perturbacija predstavlja majhno spremembo scenarija; gre za predpisan način transformiranja scenarija. Npr. določena perturbacija lahko omogoča spreminjanje uteži povezav v omrežju, druga odstranjevanje in dodajanje povezav v grafu itd. Za dani problem lahko predpišemo dovoljene ali prepovedane perturbacije. Z omejitvijo dovoljenih perturbacij lahko dosežemo lažjo računsko obvladljivost problema.

Pomembni[5] sta naslednji dve vrsti perturbacij.

Kvantitativne perturbacije. V to skupino sodijo vse perturbacije, ki omogočajo spreminjanje izključno kvantitativnih podatkov, ne omogočajo pa spreminjanja notranje strukture in odnosov med posameznimi podatki. Kvantitativno perturbirani scenariji se med seboj lahko razlikujejo le v vrednostih številskih podatkov. Sem sodi npr. spreminjanje uteži povezav omrežja.

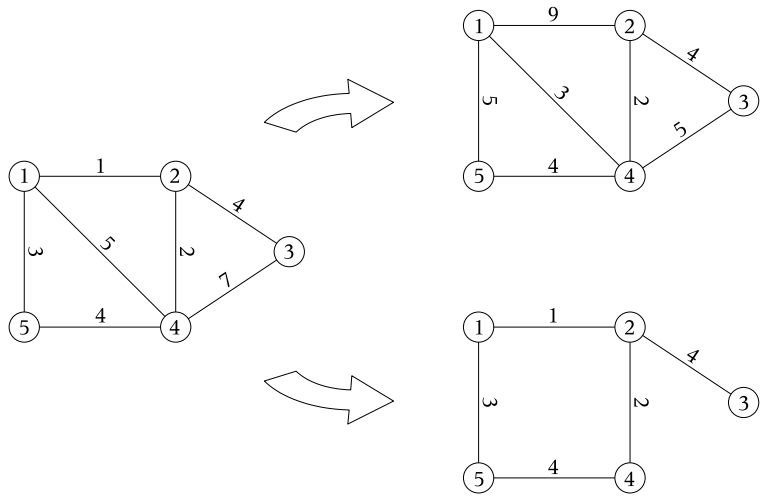
Kvalitativne perturbacije. V to skupino sodijo vse perturbacije, ki omogočajo spreminjanje izključno kvalitativnih podatkov. Gre torej za spreminjanje notranje strukture ali odnosov med posameznimi podatki. Vključujejo npr. odstranjevanje ali dodajanje povezave iz oz. v graf itd.

Pogosto kvalitativne perturbacije posredno vsebujejo tudi transformacije kvantitativnih podatkov. Npr. če odstranimo povezavo v omrežju, s tem seveda odstranimo tudi utež povezave, ki pa je kvantitativen podatek. Posledično so problemi, v katerih nastopajo kvalitativne perturbacije, pogosto težji za reševanje kot problemi, v katerih so dovoljene le kvantitativne perturbacije.

Oglejmo si primer perturbacij na sliki 2.4. Omrežje na levi strani slike predstavlja osnovni scenarij, iz katerega s pomočjo perturbacij generiramo ostale scenarije. Na desni strani slike zgoraj se nahaja scenarij, ki je pridobljen z uporabo kvantitativnih perturbacij (v tem primeru spreminjanje uteži povezav); desno spodaj pa je prikazan scenarij, pridobljen z uporabo kvalitativnih perturbacij (odstranjevanje povezav).

[5] Gre za neformalno razdelitev.

Slika 2.4: Kvantitativne in kvalitativne perturbacije



2.4.4 Robustna optimizacija

Na področje robustne optimizacije [KY97] sodi iskanje *robustne* rešitve za dane scenarije, t.j. ene same rešitve, ki je "dobra" za vse vhodne scenarije. Dopustne rešitve robustnega problema so presek dopustnih rešitev posameznih scenarijev, t.j. $S(s_1, s_2, \ldots, s_n) = \cap_i S(s_i)$. Kot optimizacijski kriterij se uporablja *mera robustnosti*; iz literature je znanih več različnih mer. Opišimo jih.

Absolutna robustnost. Rešitev te vrste je osnovana na optimizaciji najslabšega možnega primera. V problemih gre za iskanje rešitve S, ki minimizira kriterijsko funkcijo

$$z(S) = \max_i m(s_i, S).$$

Obžalovanje (ali odklon)[6]**.** Pri tem pristopu gre za iskanje rešitve S, katere odklon od optimalnosti je v najslabšem primeru najmanjši. Zapisano bolj formalno, gre za minimizacijo kriterijske funkcije

$$z(S) = \max_i \left[m(s_i, S) - m^*(s_i) \right],$$

pri tem je $m^*(s_i)$ vrednost optimalne rešitve za scenarij s_i (glede na $m(s_i, S)$).

Relativna robustnost (ali relativno obžalovanje). Pri tem pristopu gre za iskanje rešitve S, ki omogoča najmanjši relativni odklon od optimalnosti. Gre za minimizacijo kriterijske funkcije

$$z(S) = \max_i \frac{m(s_i, S) - m^*(s_i)}{m^*(s_i)}.$$

[6] *angl. regret, deviation*

Velja $z(S) = \max_i \frac{m(s_i,S)}{m^*(s_i)} - 1$, zato se kot mera relativne robustnosti pogosto uporablja tudi naslednja enakovredna definicija

$$z(S) = \max_i \frac{m(s_i, S)}{m^*(s_i)}.$$

Robustna optimizacija je dokaj novo področje, kljub temu pa se v strokovni literaturi že pojavlja [ABE05, Čib02, EBS05, Hen99, HOH, KY97, LLH05, Lin05, LMV05, MG05]. V naštetih virih gre za minimizacijo mere robustnosti, torej za optimizacijske probleme vrste *minimax*. Smiselna bi bila tudi optimizacija vrste *minisum*, vendar je v literaturi nismo zasledili, kakor tudi ne vrst *maximin* in *maxisum*.

Eden izmed razlogov za obstoj več različnih mer robustnosti je ta, da se kljub optimalnosti robustne rešitve glede na neko mero robustnosti lahko zgodi, da je robustna rešitev pravzaprav slabe kakovosti za nekatere posamezne scenarije. Oglejmo si to težavo na primeru problema najmanjši robustni 3-center z dvema scenarijema. V problemu najmanjši 3-center gre za razmestitev treh centrov v vozlišča grafa, tako da je razdalja od poljubnega vozlišča do njegovega najbližjega centra najmanjša. Oglejmo si robustni 3-center z dvema scenarijema, ki sprašuje po optimalni rešitvi glede na mero absolutne robustnosti.

Slika 2.5: Robustna rešitev.

Vhod v problem naj bosta scenarija (v tem primeru sta to omrežji), prikazana na sliki 2.5. Naj bo $m(s_i, S)$ kriterijska funkcija problema najmanjši 3-center. Optimalna robustna rešitev glede na mero absolutne robustnosti je $S = \{1, 3, 5\}$; njena vrednost je $z(S) = M$. Optimalna rešitev za vsak posamezen scenarij ima vrednost 1, t.j. $m^*(s_1) = m^*(s_2) = 1$. Vidimo, da čeprav je robustna rešitev optimalna glede na mero absolutne robustnosti, je še vedno lahko poljubno slaba za katerega od posameznih scenarijev.

2.4.5 Stohastična optimizacija

V problemih, ki sodijo na področje stohastične optimizacije, so poleg scenarijev podane tudi verjetnosti realizacije posameznih scenarijev. Glede na podane verjetnosti je cilj stohastične optimizacije poiskati rešitev, ki maksimizira pričakovano kakovost rešitve pri vseh vhodnih scenarijih. Povedano preprosteje, gre za iskanje rešitve,

ki ima večjo kakovost za bolj verjetne scenarije in morda slabšo kakovost za manj verjetne scenarije.

Obstaja več različnih pristopov iskanja stohastične rešitve. Npr. iz verjetnostnih porazdelitev parametrov lahko izračunamo pričakovani scenarij, t.j. scenarij, v katerem so vrednosti parametrov pričakovane. Za pričakovani scenarij nato na običajen deterministično način poiščemo rešitev. Gre pravzaprav za pretvorbo stohastičnega problema v determinističnega. Včasih se namesto pričakovanega scenarija uporabi kar najbolj verjeten scenarij.

Nekateri avtorji [KY97] kot slabost stohastičnega pristopa navajajo dejstvo, da je potrebno poznati verjetnosti posameznih scenarijev. Težava je v tem, da je ugotavljanje verjetnosti ali verjetnostnih porazdelitev podatkov pogosto zelo zahtevno opravilo. Včasih se celo izkaže, da so porazdelitve, pri katerih je stohastični problem obvladljiv, popolnoma neprimerne za modelirani praktični problem. In nenazadnje, poleg naštetega se lahko pojavi enaka težava kot pri robustnosti: v primeru realizacije manj verjetnega scenarija se lahko zgodi, da je najdena stohastična rešitev slabe kakovosti za dejansko realizirani scenarij.

2.5 Pretoki in ujemanja

2.5.1 Pretoki v omrežju

Definicije

Zelo pomembni in uporabni optimizacijski problemi so problemi iskanja različnih vrst pretokov v omrežju. Pogosto se primeri, da je nek praktični problem možno preprosto modelirati s problemom pretokov, kakor tudi, da je nek optimizacijski problem učinkovito rešljiv s pomočjo kakega od problemov pretokov. V ta namen bomo probleme pretokov uporabljali tudi mi, zato jih v tem razdelku na kratko predstavimo. Pomembnejša literatura o tem področju je [CLRS01, KV00, PS98, Vil02].

Problemi iskanja različnih pretokov so definirani na omrežju $G = (V, E)$, kjer je vsaki povezavi $(u, v) \in E$ prirejena *cena* $w_{uv} \in \mathbb{R}$ in *kapaciteta* $c_{uv} \in \mathbb{R}$. Cena povezave navadno predstavlja ceno na enoto pretoka. Kapaciteta povezave je največji možni pretok po povezavi. Naj f_{uv} predstavlja pretok skozi povezavo $(u, v) \in E$. Tedaj očitno velja

$$0 \leq f_{uv} \leq c_{uv}.$$

V grafu G sta dve posebni vozlišči, *izvor* $s \in V$ in *ponor* $t \in V$. Za vsako vozlišče $v \in V \setminus \{s, t\}$ velja *zakon o ohranitvi pretoka*

$$\sum_{u \in V} f_{uv} = \sum_{u \in V} f_{vu}.$$

Leva stran enačbe predstavlja *pritok* v vozlišče v in desna *iztok* iz v; celotna enačba torej pomeni, da kar priteče v vozlišče v, mora iz njega tudi izteči.

Dopustna rešitev (f_{uv}) problema iskanja pretoka je nabor vrednosti $f_{uv} \in \mathbb{R}$, ki ustreza zgornjima pogojema, t.j. $(f_{uv}) = \{f_{uv} | (u, v) \in E\}$. *Vrednost* pretoka (f_{uv}) je definirana kot

$$|(f_{uv})| = \sum_{v \in V} f_{sv} - \sum_{v \in V} f_{vs} = \sum_{v \in V} f_{vt} - \sum_{v \in V} f_{tv}.$$

Vrednost pretoka je torej enaka neto iztoku izvora oz. neto pritoku ponora. Pogosto bomo tako dopustni rešitvi kot vrednosti pretoka rekli kar *pretok*.

Problemi pretokov

Iskanje največjega pretoka skozi omrežje je definirano kot naslednji problem.

Problem 2.1: Največji Pretok

Naloga: Omrežje $G = (V, E)$, kjer je za vsako povezavo $(u, v) \in E$ podana kapaciteta $c_{uv} \geq 0$. Podana sta še izvor $s \in V$ in ponor $t \in V$.
Dopustna rešitev: Pretok (f_{uv}) iz s v t.
Kakovost dopustne rešitve: $|(f_{uv})|$.
Cilj: Maksimizacija.

Za Največji Pretok obstaja več hitrih algoritmov, ki jih navajamo hkrati z njihovimi časovnimi zahtevnostmi v tabeli 2.6. Za algoritem Ford-Fulkersona je

Tabela 2.6: Algoritmi za Največji Pretok

Algoritem	Časovna zahtevnost
Ford-Fulkerson	$O(\|E\|\|(f_{uv})\|)$
Edmons-Karp	$O(\|V\|\|E\|^2)$
Dinic	$O(\|E\|\|V\| \log \|V\|)$
Goldberg	$O(\|V\|^2\|E\|)$
Goldberg-Tarjan	$O(\|V\|^3)$

časovna zahtevnost podana za naloge, v katerih so kapacitete povezav cela števila; v nasprotnem primeru algoritem pravzaprav sploh ne zagotavlja ustavljivosti in optimalnosti. Izboljšana različica je Edmons-Karpov algoritem, ki je bil poleg Dinicevega algoritma prvi polinomski algoritem za Največji Pretok.

Če so v nalogi problema pretokov kapacitete povezav celoštevilske, potem obstaja celoštevilski največji pretok. To dejstvo izhaja že iz samega postopka Ford-Fulkersonovega algoritma. Zapišimo to v obliki naslednje trditve.

Lema 2.5 ▶ *Če v $G = (V, E)$ za vse $(u, v) \in E$ velja $c_{uv} \in \mathbb{Z}^+$, potem v G obstaja največji pretok vrednosti $|(f_{uv})| \in \mathbb{Z}^+$. Še več, za vsak $(u, v) \in E$ velja $f_{uv} \in \mathbb{Z}^+$.*

Poznamo še uteženo različico iskanja pretoka skozi omrežje. Gre za problem iskanja najcenejšega pretoka vrednosti p, ki je podan z naslednjo definicijo.

Problem 2.2: Najmanjša Cena Pretoka

Naloga: Omrežje $G = (V, E)$, kjer je za vsako povezavo $(u, v) \in E$ podana njena cena $w_{uv} \geq 0$ in kapaciteta $c_{uv} \geq 0$. Podani so še ciljna vrednost celotnega pretoka p, izvor $s \in V$ in ponor $t \in V$.
Dopustna rešitev: Pretok (f_{uv}) iz s v t z vrednostjo p.
Kakovost dopustne rešitve: $\sum_{(u,v)\in E} f_{uv} w_{uv}$.
Cilj: Minimizacija oz. odločitev, da rešitev ne obstaja.

Gre torej za minimizacijo celotne cene pretoka, ki je definirana kot vsota po vseh povezavah $(u, v) \in E$ produktov cen w_{uv} in pretokov f_{uv}. Obstoj rešitve za problem Najmanjša Cena Pretoka je opisan z naslednjo trditvijo.

Lema 2.6 ▶ *Če v G obstaja največji pretok vrednosti $\geq p$, potem v G obstaja tudi (najcenejši) pretok vrednosti p.*

V primeru obstoja pretoka vrednosti p v G lahko tak pretok poiščemo tako, da v G dodamo vozlišče s' in povezavo kapacitete p iz s' v s; v tem omrežju pa poženemo algoritem za Največji Pretok.

Za problem Najmanjša Cena Pretoka obstaja več različnih algoritmov, dva pomembnejša sta skupaj s časovnimi zahtevnostmi podana v tabeli 2.7. Obstoj prvega algoritma pravzaprav dokazuje, da je Najmanjša Cena Pretoka v razredu $\mathcal{P}$. Drugi algoritem je v primeru, da $p = O(|V|)$, asimptotično najhitrejši algoritem.

Tabela 2.7: Algoritmi za Najmanjša Cena Pretoka

Algoritem	Časovna zahtevnost
Najkrajši Povprečni Cikel[7]	$O(\|E\|^3\|V\|^2 \log \|V\|)$
Zaporedne Najkrajše Poti[8]	$O(\|V\|\|E\| + p(\|V\|^2 + \|E\|))$

Zapišimo še naslednjo posledico.

Posledica 2.7 ▶ *Če $p \in \mathbb{Z}^+$ in če v $G = (V, E)$ za vse $(u, v) \in E$ velja $c_{uv} \in \mathbb{Z}^+$, potem za najcenejši pretok vrednosti p (če seveda tak pretok obstaja) za vsak $(u, v) \in E$ velja $f_{uv} \in \mathbb{Z}^+$.*

[7] *angl. Minimum-mean-cycle-cancelling*

[8] *angl. Successive Shortest-Paths.* Algoritem deluje, če so p in kapacitete povezav celoštevilčni [KV00].

2.5.2 Ujemanja

Problemi, ki se velikokrat pojavljajo v praksi in ki jih lahko učinkovito rešimo s pomočjo problemov pretokov, so problemi iskanja različnih ujemanj[9] v grafu. Tudi te bomo potrebovali v nadaljevanju. Najprej povejmo, kaj je ujemanje.

Definicija 2.1: Ujemanje, maksimalno in popolno ujemanje

Naj bo $G = (V, E)$ graf. Množica $M \subseteq E$, v kateri noben par povezav nima skupnega krajišča, se imenuje *ujemanje*. Ujemanje, ki ni podmnožica kakega drugega ujemanja, se imenuje *maksimalno ujemanje*. Maksimalno ujemanje, katerega krajišča povezav tvorijo vsa vozlišča V, se imenuje *popolno ujemanje*.

Maksimalno ujemanje je ujemanje, kateremu ne moremo dodati nobene povezave več, da bi še vedno bilo ujemanje. Vsa ujemanja niso maksimalna in vsa maksimalna ujemanja niso popolna. Najprej definirajmo problem iskanja največjega ujemanja v grafu.

Problem 2.3: Največje Ujemanje

Naloga: Graf $G = (V, E)$.
Dopustna rešitev: Ujemanje $M, M \subseteq E$, v G.
Kakovost dopustne rešitve: $|M|$.
Cilj: Maksimizacija.

Bolj nas bo zanimala različica problema Največje Ujemanje, v kateri je graf G dvodelen. To različico lahko zelo učinkovito rešujemo s prevedbo na Največji Pretok. Časovna zahtevnost takega algoritma je $O(|V||E|)$.

Seveda lahko definiramo še problem, v katerem povezavam dodamo še ceno. Tako dobimo naslednji problem iskanja ujemanja z največjo ceno.

Problem 2.4: Največja Cena Ujemanja

Naloga: Omrežje $G = (V, E)$, kjer je utež $w(e)$ cena povezave $e \in E$.
Dopustna rešitev: Ujemanje $M, M \subseteq E$, v G.
Kakovost dopustne rešitve: $\sum_{e \in M} w(e)$.
Cilj: Maksimizacija.

Problem Največja Cena Ujemanja je pravzaprav enakovreden naslednjemu problemu iskanja najcenejšega popolnega ujemanja [KV00].

[9] Včasih se za ujemanje uporablja tudi izraz prirejanje.

Problem 2.5: Najmanjša Cena Popolnega Ujemanja

Naloga: Omrežje $G = (V, E)$, kjer je utež $w(e)$ cena povezave $e \in E$.
Dopustna rešitev: Popolno ujemanje M, $M \subseteq E$, v G.
Kakovost dopustne rešitve: Cena ujemanja M, t.j. $\sum_{e \in M} w(e)$.
Cilj: Minimizacija.

V nadaljevanju bo za nas posebej zanimiva različica problema, v kateri je vhodni graf G dvodelen. Ta preprostejša različica ima celo svoje ime in se imenuje Najmanjša Dodelitev. Problem Najmanjša Dodelitev lahko učinkovito rešimo s problemom Najmanjša Cena Pretoka. Časovna zahtevnost takega algoritma je $O(|V|^3)$.

Poglavje 3

Prilagodljivost

Vse teče, nič ne miruje.
– Heraklit

Poglavje najprej predstavlja prilagodljivost s splošnega vidika. Povemo, zakaj je raziskovanje prilagodljivosti pomembno, opišemo način vrednotenja uspešnosti prilagajanja in naštejemo pristope, sorodne prilagajanju. Nato podrobno opišemo način vpeljave prilagodljivosti v optimizacijske probleme. Vpeljemo nekaj novih pojmov, kot so: partikularne rešitve, partikularna kakovost, graf prehodov scenarijev itd. Na koncu predstavimo še nekaj možnih različic prilagodljivosti.

3.1 Prilagodljivost na splošno

3.1.1 Prilagodljivost in prilagajanje

Preden se lotimo natančnega obdelovanja *prilagodljivosti v optimizacijskih problemih*, si na kratko oglejmo pomen izraza prilagodljivost. V SSKJ[1] [ABG+98] pod tem in sorodnimi gesli najdemo vrsto vsebinsko podobnih opisov. Pomen vseh izhaja predvsem iz gesla *prilagajati*, ki je definiran kot "delati, da kaj dobi potrebne lastnosti, značilnosti glede na lastnosti, značilnosti česa" ali kot "usklajevati svoje vedenje, ravnanje, mišljenje s kom, čim". Zanimivejši primeri uporabe so: "naselja je treba prilagajati pokrajini, živa bitja se prilagajajo spremenjenim življenjskim razmeram, prilagajati zakone novim razmeram, radio in televizija morata prilagajati svoj program poslušalcem" itd.

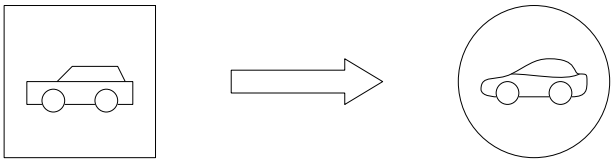
Skušajmo ostati kar se da splošni. Poimenujmo tisto, kar prilagajamo, *objekt* in tisto, čemur objekt prilagajamo, *okolje*. *Prilagajanje* je torej ustrezno spreminjanje, obdelovanje in/ali usklajevanje lastnosti in značilnosti nekega objekta glede na njegovo okolje. *Prilagodljivost* je sposobnost objekta za prilagajanje in rezultat oz. posledica prilagajanja je *prilagoditev* objekta.

[1] Slovar slovenskega knjižnega jezika.

V idealnem svetu bi bilo prilagajanje brezplačno, brez porabe časa, surovin ali kakršnihkoli drugih sredstev. A v realnosti praviloma ni tako; za prilagajanje je vedno potrebno plačati neko ceno. V nadaljevanju se bomo ukvarjali izključno s takšnimi (ne brezplačnimi) prilagajanji. Naš cilj bo poiskati objekt, za prilagajanje katerega bo cena najmanjša.

Osvetlimo povedano s primerom podjetja, ki se ukvarja z oblikovanjem avtomobilov. Znano je, da oblikovanje sledi modnim smernicam, ki se iz leta v leto nenehno spreminjajo. Na sliki 3.1 je prikazano, kako bi se lahko oblika avtomobila spremenila pri prehodu iz "oglatega" modnega stila v "zaobljenega". V tem primeru je

Slika 3.1: Primer prilagajanja.



avtomobil objekt prilagajanja, kvadrat predstavlja staro okolje in krog novo okolje. Cilj pa je stari oglati avto na najcenejši način spremeniti v novega zaobljenega.

Navedeni primer prikazuje prilagajanje kot dogodek, ki se je zgodil samo enkrat, vendar na splošno ni tako. Okolje se lahko spremeni večkrat, vsaka sprememba okolja pa lahko vodi do prilagajanja objekta. Prilagajanje je torej običajno trajajoč in kontinuiran proces; ko in če se okolje spremeni, se sproži prilagajanje objekta.

Iskanje optimalne prilagoditve objekta novemu okolju se seveda lahko sproži tudi v trenutku ob spremembi okolja. S tem se ukvarja področje realno-časovnih[2] sistemov in procesov, s katerimi pa se v nadaljevanju ne bomo ukvarjali. Osredotočili se bomo predvsem na procese, v katerih je spremembe okolja možno *načrtovati* in / ali *napovedovati*. Popolno načrtovanje sprememb okolja zaradi nedoločenosti večkrat ni možno, zato v takšnem primeru skušamo spremembe okolja napovedati. S samim načrtovanjem in napovedovanjem sprememb okolja se ne bomo ukvarjali; namesto tega bomo predpostavljali, da so spremembe okolja podane kot vhodni podatek problema.

V tem razdelku smo osvetlili samo vsebino pojma prilagodljivost, v naslednjem pa si oglejmo še nekaj razlogov za njeno proučevanje.

3.1.2 Pomembnost prilagodljivosti

Ljudje, organizacije, podjetja, procesi, metode itd. so dandanes podvrženi naglim spremembam. Spreminjajo se zakoni, pogoji poslovanja, načini distribuiranja, nastajajo nova znanja in tehnologije itd. Oglejmo si nekaj primerov različnih razlogov, zaradi katerih se je tem spremembam potrebno prilagajati.

[2] *angl. real-time*

V Sloveniji smo bili pred leti priča davčni reformi, s katero se je obstoječi davek od prometa nadomestil z davkom na dodano vrednost. Podjetja, ki so hotela neprekinjeno delovati na trgu, so morala svoj način poslovanja prilagoditi novemu načinu vodenja davčnih obveznosti, saj bi bilo sicer njihovo nadaljnje obratovanje nezakonito. Torej so se podjetja morala prilagoditi zaradi nuje, t.j. bolj pomembno je bilo pravilno vodenje davčnih obveznosti, kot pa cena preoblikovanja starega načina v novega.

Kot nekakšno nasprotje si oglejmo primer iz avtomobilske industrije. Tipičen današnji avtomobil je načrtovan tako, da ga je možno prilagoditi različnim zahtevam trgov in potrebam voznikov, kot so npr. vožnja po levi ali desni strani voznega pasu, vgradnja motorjev različnih zmogljivosti, različna izbira notranje opreme itd. Proizvajalčeva želja je seveda zasnovati takšen model avtomobila, katerega prilagajanje na različne zahteve in potrebe bo kar se da poceni. Zaradi boja s tekmeci je v poslovnem svetu večkrat bolj pomembno doseči nižje stroške prilagajanja, kot pa popolno zasnovo avtomobila za vsako od zahtev. Gre za prilagodljivost zaradi dobička, zato lahko kakovost izdelka trpi zaradi prilagodljivosti.

Večkrat imamo prilagodljivost za lastnost uspešnega podjetja, kakovostnih izdelkov ali storitev. To lahko hitro preverimo, če v enega od spletnih iskalnikov vnesemo geslo "prilagodljivost"[3]. Kot rezultat takšnega iskanja dobimo ogromno povezav na spletne strani, na katerih podjetja hvalijo prilagodljivost svojih izdelkov ali storitev. Izraz prilagodljivost ima očitno močan pozitiven prizvok, zato je večkrat izkoriščan za potrebe trženja. Seveda mora podjetje prilagodljiv izdelek ali storitev, preden ga v resnici ponudi, izdelati. Gre za nekakšno prilagodljivost zaradi boljšega trženja.

Internet in druge podobne sodobne telekomunikacijske tehnologije so v zadnjem času omogočile razmah globalizacije. Globalna podjetja si prizadevajo za širino svojega delovanja in prisotnost na svetovnem trgu. S tem naletijo na nove razmere, ki jim morajo prilagoditi svoje izdelke ali storitve. Kot primer navedimo tri podjetja, ki svoje izdelke oz. storitve tržijo izključno preko svetovnega spleta. Njihove spletne strani so prilagojene različnim jezikovnim področjem. Tako ima (v času pisanja disertacije) spletna trgovina Amazon.com[4] spletno stran v 7 jezikih, letalska prevoznika RyanAir[5] in EasyJet[6] pa celo v 20 oz. 15 jezikih. Podjetja se morajo prilagajati zaradi širine svojega delovanja.

3.1.3 Splošnost prilagodljivosti

Ugotovili smo, da imamo precej razlogov za proučevanje prilagodljivosti, poleg tega pa tudi, da je prilagodljivost širok pojem, ki za natančno proučevanje potrebuje jasno definicijo. Vpeljave prilagodljivosti v optimizacijske probleme se bomo podrobno lotili v naslednjem razdelku, še pred tem pa v tem podrazdelku naštejmo nekaj že

[3] *angl. flexibility, adaptability*
[4] http://www.amazom.com
[5] http://www.ryanair.com
[6] http://www.easyjet.com

znanih pristopov s področja optimizacije, ki so v marsičem sorodni prilagajanju oz. na katere lahko pogledamo skozi prizmo prilagodljivosti.

Optimizacija. Kadar v problemu prilagajanja nastopa samo en objekt in eno okolje, je prilagajanje kar "običajna" optimizacija, kakršne smo že dobro vajeni. S takim prilagajanjem se v nadaljevanju ne bomo ukvarjali. Ker pa je samo prilagajanje pravzaprav optimizacijski problem, bo področje optimizacije za nas predstavljalo raziskovalno osnovo.

Realno-časovni sistemi. Gre za sisteme z zelo kratkim odzivnim časom na spremembe v okolju. Spremembe niso načrtovane; takšen sistem se praktično takoj po spremembi odzove in prilagodi svoje delovanje. Značilnost realno-časovnih sistemov je poleg hitre odzivnosti navadno tudi velika zanesljivost delovanja, zato so takšni sistemi dragi in se pogosto uporabljajo v aplikacijah, kjer gre za varnost človeškega življenja, kot npr. nadzor vitalnih funkcij pacientov, navigacijski sistem v letalih, protiblokirni zavorni sistem v avtomobilih itd.

Online algoritmi. Algoritem, ki vhodne podatke obdeluje po delih, ne da bi jih imel v celoti na voljo od samega začetka, se imenuje *online* (sprotni) algoritem. V nasprotju s tem so pri *offline* algoritmu vhodni podatki v celoti na voljo ob zagonu algoritma. Količina razpoložljivih vhodnih podatkov online algoritma narašča med njegovim izvajanjem. Kot primer si lahko bralec ogleda online algoritem za problem razvrščanja poslov v [Rob02]. Pregledno delo s tega področja je [BEY98].

Dinamičnost. Problemom, v katerih so podatki odvisni od časa, pravimo tudi dinamični problemi. Pri njih gre večkrat za sosledje dogodkov, pri čemer je časovna odvisnost podatkov znana vnaprej. Potrebno pa je poiskati rešitev, ki je primerna za vse časovne trenutke. Več o vključevanju dinamičnosti v optimizacijske probleme obravnava [Čib02], primere dinamičnih problemov pa [BGKS98, HP98, SM96].

Optimizacija scenarijev. Scenariji se uporabljajo za modeliranje nedoločenosti vhodnih podatkov, pri čemer scenarij predstavlja možno realizacijo vhodnih podatkov. Več scenarijev skupaj predstavlja vhod v problem optimizacije scenarijev, katerega cilj je poiskati rešitev, ki je na nek način primerna za vse podane scenarije. Ker v teh problemih nastopa več scenarijev, se uporabljajo različni kriteriji vrednotenja kakovosti rešitve. Poznana sta predvsem robustnostni in stohastičnostni kriterij. Več o optimizaciji scenarijev je v [KY97, Sny04].

Robustnost in stohastičnost. Z vidika optimizacije scenarijev je robustna rešitev takšna rešitev, ki je primerna za vse podane scenarije, stohastična pa takšna, ki je bolj primerna za bolj verjetne scenarije in manj primerna za manj verjetne scenarije. Oba kriterija lahko posplošimo tudi izven področja optimizacije

scenarijev, kar pomeni, da ni nujno, da so potencialne situacije predstavljene s scenariji. Pregledno delo s področja robustnosti je [KY97], več o stohastičnosti je v [SS].

Mobilnost ponudnikov. Oglejmo si problem razmeščanja mobilnih ponudnikov, katerega vhodni podatki so različni scenariji. Za vsak scenarij je potrebno poiskati razmestitev nekega števila ponudnikov, pri čemer pa se lahko razmestitve spreminjajo glede na scenarij. Optimizacijski kriterij je lahko npr. maksimizacija števila fiksnih (s stalno lokacijo) ponudnikov.

3.2 Prilagodljivost v optimizacijskih problemih

V tem razdelku se bomo podrobneje lotili prilagodljivosti v optimizacijskih problemih. Opisali bomo, kako iz nekega optimizacijskega problema ustvarimo njegovo prilagodljivo različico. Potrudili se bomo, da bo vpeljava prilagodljivosti v optimizacijske probleme kar se da splošna, vendar hkrati dovolj jasno in matematično definirana, da bo omogočala nadaljnjo podrobno obravnavo. V naslednjih nekaj podrazdelkih bomo postopoma predstavili sestavine prilagodljivostnega optimizacijskega problema. Na koncu vsakega izmed podrazdelkov bomo opisali še praktični primer, ki bo prikazal uporabo novih pojmov.

3.2.1 Vpeljava prilagodljivosti

Scenariji in partikularne rešitve

Naj bo $P = (I, S, m, cilj)$ optimizacijski problem, kjer je I množica nalog problema P, S je funkcija, ki vsaki nalogi $x \in I$ priredi množico $S(x)$ dopustnih rešitev, m je kriterijska funkcija, ki vsaki dopustni rešitvi $y \in S(x)$ naloge x priredi vrednost $m(x, y) \in \mathbb{R}$ in $cilj \in \{\min, \max\}$, ki pove, ali je potrebno poiskati rešitev z najmanjšo ali največjo vrednostjo kriterijske funkcije.

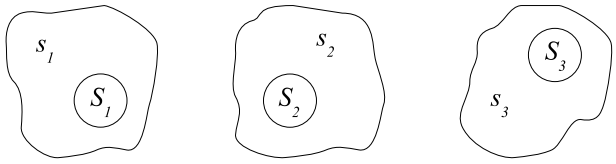
Iz problema P, ki mu bomo včasih rekli tudi *osnovni optimizacijski problem*, bomo s pomočjo *scenarijev* v nekaj naslednjih podrazdelkih izpeljali *prilagodljivostni optimizacijski problem* $P' = (I', S', m', cilj')$. Nalogo $x' \in I'$ problema P' sestavlja množica scenarijev. Spomnimo, da scenarij predstavlja nek nabor vhodnih podatkov, kar s stališča problema P ustreza nalogi $x \in I$. Naloga $x' \in I'$ problema P' je torej sestavljena iz množice scenarijev $\{s_1, s_2, \ldots, s_n\} \subseteq I$.

Glavna ideja prilagodljivosti pa je v tem, da za vsakega od scenarijev s_i poiščemo "ustrezno" rešitev (s stališča osnovnega optimizacijskega problema), tako da so si dobljene rešitve čimbolj "podobne". (Več o ustreznosti in podobnosti v nadaljevanju.) V idealnem primeru so vse rešitve enake – s tem se ukvarjata robustnost in stohastičnost – vendar to v praksi ni vedno uresničljivo. Zato bomo v nadaljevanju predpostavljali, da so vsaj nekatere izmed posameznih rešitev različne. Rešitvi sce-

narija s_i bomo rekli *partikularna rešitev* in jo označili z S_i. Vse partikularne rešitve skupaj sestavljajo *celotno rešitev* naloge $x' \in I'$ problema P'.

Primer treh scenarijev s_1, s_2 in s_3 ter njihovih pripadajočih partikularnih rešitev S_1, S_2 in S_3 je prikazan na sliki 3.2.

Slika 3.2: Scenariji in njihove partikularne rešitve.



Dopustna rešitev naloge $x' \in I'$ problema P' je množica $\{S_1, S_2, \ldots, S_n\}$, kjer je S_i *dopustna partikularna rešitev* scenarija s_i. (Dopustne partikularne rešitve pa so pravzaprav dopustne rešitve naloge, ki jo dobimo, če na scenarij neodvisno pogledamo kot na nalogo problema P.)

Primer. Podjetje razmišlja o začetku proizvodnje in distribucije nove vrste izdelkov. Zgraditi namerava več ponudnikov (t.j. proizvodnih obratov in distribucijskih centrov) v različnih regijah sveta. V vsaki regiji ima na voljo več lokacij za postavitev ponudnikov. Podjetje se odloči, da bo v vsaki regiji postavilo $p \geq 1$ ponudnikov. V tem primeru scenarij s_i predstavlja množico vseh potencialnih lokacij v določeni regiji, partikularna rešitev S_i pa izbor p izmed njih. Pri tem je izbor ponudnikov odvisen od omejitev in kriterijev, ki nastopajo v problemu.

Kriterij vrednotenja rešitev

Glede na povedano je prav gotovo na mestu vprašanje, kakšen je kriterij vrednotenja rešitev naloge $x' \in I'$ prilagodljivostnega optimizacijskega problema. Glede na zgoraj omenjeni ustreznost in podobnost partikularnih rešitev se porajata naslednja dva vidika.

Kakovost. Ustreznost partikularne rešitve S_i scenariju s_i bomo natančneje definirali kot *partikularno kakovost*. Slednja bo neposredno izpeljana iz kriterijske funkcije problema P. Več o kakovosti bomo povedali v podrazdelku 3.2.2.

Prilagodljivost. Podobnost dveh partikularnih rešitev pa bomo natančneje definirali kot *partikularno prilagodljivost*. V povezavi s tem bomo večkrat govorili tudi o ceni medsebojnega prilagajanja partikularnih rešitev, kjer gre za ceno transformacije ene partikularne rešitve v drugo. Več o prilagodljivosti bomo povedali v podrazdelku 3.2.3.

Kriterija kakovosti in prilagodljivosti največkrat nista neodvisna. Povečevanje enega lahko vpliva na zmanjševanje drugega, zaradi česar je hkratna optimizacija

obeh kriterijev lahko precej zamotano opravilo. Nastalo zagato je možno reševati na različne načine, od katerih omenimo večkriterijsko optimizacijo ali uporabo kriterija kakovosti kot omejitve v problemu. Pri modeliranju prilagodljivostnih optimizacijskih problemov je zato večkrat potrebno ubrati kompromisno pot; posledično je pri tem večkrat vpleten človeški faktor.

3.2.2 Kakovost

Partikularna kakovost

Nalogo prilagodljivostnega optimizacijskega problema sestavlja množica scenarijev, pri čemer za vsakega od njih poiščemo partikularno rešitev. Najprej si oglejmo *partikularno kakovost*. Ustreznost partikularne rešitve S_i scenariju s_i bomo poimenovali *partikularna kakovost* in jo definirali kot

$$z_i(S_i) = m(s_i, S_i),$$

kjer je $m(x, y)$ kriterijska funkcija osnovnega optimizacijskega problema. Partikularna kakovost pravzaprav predstavlja pogled na kakovost rešitve S_i scenarija s_i neodvisno od (brez upoštevanja) ostalih scenarijev.

Označimo z S_i^* optimalno rešitev scenarija s_i glede na z_i. (Gre za optimalno rešitev naloge s_i problema P.) Z vidika problema P' pa bomo S_i^* imenovali *optimalna partikularna rešitev*. Definirajmo še *optimalno partikularno kakovost* z_i^* kot partikularno kakovost optimalne partikularne rešitve, t.j. $z_i^* = z_i(S_i^*)$.

Celotna kakovost

Na osnovi partikularnih kakovosti lahko definiramo *celotno kakovost*, s katero merimo kakovost celotne rešitve. Celotno kakovost bomo označili z

$$z(S_1, S_2, \ldots, S_n),$$

pri čemer je S_i partikularna rešitev za scenarij s_i, kjer $1 \leq i \leq n$.

Funkcijo $z(S_1, S_2, \ldots, S_n)$ lahko seveda izračunamo na različne načine, v nadaljevanju pa bo to največkrat kar vsota, maksimum ali minimum partikularnih kakovosti. V primeru vsote je torej

$$z_{sum}(S_1, S_2, \ldots, S_n) = \sum_{1 \leq i \leq n} z_i(S_i),$$

v primeri maksimuma

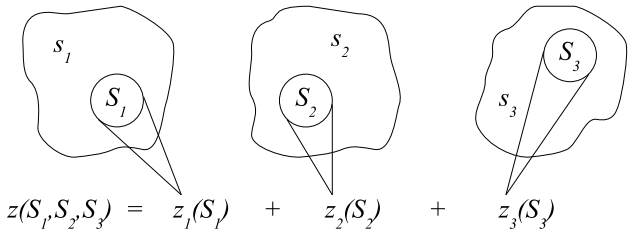
$$z_{max}(S_1, S_2, \ldots, S_n) = \max_{1 \leq i \leq n} z_i(S_i)$$

in v primeru minimuma

$$z_{min}(S_1, S_2, \ldots, S_n) = \min_{1 \leq i \leq n} z_i(S_i).$$

Na sliki 3.3 je prikazan primer treh scenarijev $\{s_1, s_2, s_3\}$, njihovih partikularnih rešitev $\{S_1, S_2, S_3\}$ in partikularnih kakovosti z_1, z_2, z_3, kakor tudi celotna kakovost $z(S_1, S_2, S_3)$, ki je v danem primeru vsota partikularnih kakovosti.

Slika 3.3: Partikularna in celotna kakovost.



Kakovost v prilagodljivostnem optimizacijskem problemu

Kriterijska funkcija prilagodljivostnega optimizacijskega problema je lahko sestavljena iz celotne ali tudi partikularne kakovosti. Obe pa lahko v problemu nastopata tudi kot omejitev. Oglejmo si najprej partikularno kakovost kot omejitev. V primeru minimizacijskega problema P' so lahko takšne omejitve oblike

$$z_i(S_i) \leq \alpha_i z_i^*,$$

kar pomeni, da je lahko partikularna kakovost $z_i(S_i)$ kvečjemu α_i-krat slabša od optimalne partikularne kakovosti z_i^*.

Če je optimalne partikularne rešitve možno poiskati v polinomskem času, t.j. problem $P \in \mathcal{P}$, potem je smiselno zahtevati $\alpha_i = 1$. Če pa je P $\mathcal{NP}$-težek problem, lahko s spreminjanjem α_i nastavljamo željeno kakovost partikularnih rešitev. Seveda v tem primeru s spreminjanjem α_i ne smemo pretiravati, da ne presežemo aproksimacijskega praga problema.

Podobno lahko tudi celotna kakovost v prilagodljivostnem optimizacijskem problemu nastopa kot omejitev. Zopet za primer minimizacijskega problema P' je takšna omejitev lahko oblike

$$z(S_1, S_2, \ldots, S_n) \leq \alpha z^*,$$

kjer je $z^* = z(S_1^*, S_2^*, \ldots, S_n^*)$ celotna kakovost optimalnih partikularnih rešitev $S_1^*, S_2^*, \ldots, S_n^*$ in α parameter, s katerim nastavljamo celotno kakovost rešitve.

Primer. Podjetje (iz primera v prejšnjem razdelku) ima v vsaki od regij na voljo več možnih lokacij za postavitev ponudnikov. Poleg tega je v vsaki regiji več že znanih lokacij porabnikov (trgovskih centrov, trgovin, končnih kupcev itd.). V tem primeru lahko partikularno kakovost, kjer partikularna rešitev S_i predstavlja izbor p izmed potencialnih lokacij v s_i za postavitev ponudnikov, izračunamo kot kriterijsko funkcijo katerega od že znanih problemov razmeščanja, npr. Najmanjši p-center ali Najmanjša k-mediana. Izbran problem razmeščanja torej predstavlja osnovni optimizacijski problem, iz katerega gradimo njegovo prilagodljivostno različico.

3.2.3 Prilagodljivost

Partikularna prilagodljivost

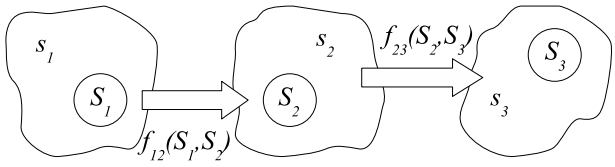
Oglejmo si še drugi kriterij, ki pravzaprav tvori bistvo prilagodljivosti v optimizacijskih problemih, zato tudi nosi enako ime, t.j. *prilagodljivost*. Kot smo že omenili, je cilj prilagodljivostnega optimizacijskega problema poiskati podobne partikularne rešitve za posamezne scenarije. Podobnost dveh partikularnih rešitev bomo poimenovali *partikularna prilagodljivost* in jo označili z

$$f_{ij}(S_i, S_j),$$

kjer sta S_i in S_j dve partikularni rešitvi za scenarija s_i in s_j zaporedoma. Velja omeniti, da se funkcija f_{ij} v splošnem razlikuje glede na scenarija s_i in s_j, ki v njej nastopata.

Funkcija $f_{ij}(S_i, S_j)$ predstavlja *ceno transformiranja oz. prilagajanja* S_i v S_j in pomeni ceno, ki jo je potrebno plačati, če pride do prehoda iz scenarija s_i v s_j. Strogo gledano pravzaprav velja, da večja kot je cena prilagajanja dveh partikularnih rešitev, manjša je njuna partikularna prilagodljivost in obratno. V nadaljevanju bomo oba pojma izmenično uporabljali.

Slika 3.4: Partikularna prilagodljivost.



Na sliki 3.4 so prikazani trije scenariji $\{s_1, s_2, s_3\}$ in njihove partikularne rešitve $\{S_1, S_2, S_3\}$. Poleg tega sta prikazani še partikularni prilagodljivosti $f_{12}(S_1, S_2)$ in $f_{23}(S_2, S_3)$. Med n scenariji lahko naredimo največ $n(n-1)$ različnih usmerjenih prehodov, vendar v realnem problemu ni nujno, da želimo upoštevati prav vse možne prehode, ampak samo nekatere. Katere prehode želimo upoštevati in katerih ne, pa lahko najlažje in najbolj splošno opišemo z grafom.

Graf prehodov scenarijev

Graf, s katerim bomo opisali, kateri prehodi med scenariji v prilagodljivostnem optimizacijskem problemu nastopajo in kateri ne, bomo poimenovali *graf prehodov scenarijev*. Če bo v modeliranem problemu smer prehodov pomembna, bo tak graf usmerjen, sicer bo neusmerjen. Naj bo torej $G = (V, E)$ enostaven in povezan (usmerjen ali neusmerjen) graf z vozlišči V in povezavami E, kjer je vozlišču $i \in V$ prirejen scenarij s_i. Graf G je graf prehodov scenarijev. Zaradi enolične povezanosti vozlišč in scenarijev bomo v nadaljevanju, kadar bomo govorili o vozlišču i, večkrat hkrati imeli v mislih scenarij s_i in tudi obratno.

Graf prehodov scenarijev je enostaven, t.j. nima zank; s tem preprečimo nesmiselna prilagajanja partikularne rešitve sami sebi. Poleg tega je tudi povezan, kar pomeni, da prilagajanje neposredno ali posredno zajema prav vse partikularne rešitve hkrati. Deli grafa, ki med seboj niso povezani, vsak zase predstavljajo samostojne probleme prilagajanja; npr. izolirane scenarije lahko obravnavamo samostojno kot osnoven optimizacijski problem.

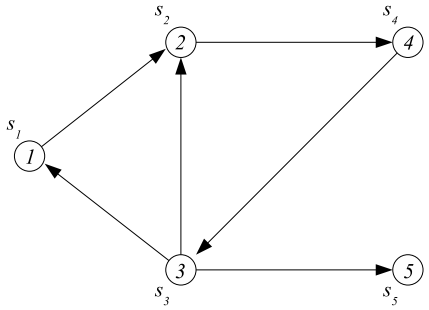
Funkcija f_{ij} je definirana za vse $(i, j) \in E$ in ni definirana za $(i, j) \notin E$. To lahko razširimo, da postavimo $f_{ij} = 0$ za vse $(i, j) \notin E$.

Naj bo $(i, j) \in E$. Če je graf G usmerjen, potem v problemu nastopa prehod iz scenarija s_i v s_j, katerega cena je $f_{ij}(S_i, S_j)$. Cena takšnega prehoda je lahko *simetrična*, t.j.

$$f_{ij}(S_i, S_j) = f_{ji}(S_j, S_i),$$

kar pomeni, da je cena prehoda v smeri od s_i proti s_j enaka kot v smeri od s_j proti s_i. V primeru neusmerjenega grafa G pa $(i, j) \in E$ pomeni prehod med s_i in s_j v poljubni smeri. Cena takšnega prehoda je vedno simetrična.

Slika 3.5: Graf prehodov scenarijev.



Na sliki 3.5 je prikazan primer grafa prehodov scenarijev s petimi vozlišči oz. scenariji.

V nadaljevanju se bomo večkrat ukvarjali s problemi, v katerih razlikovanje med usmerjenim in neusmerjenim grafom ni tako pomembno. Večkrat lahko neusmerjeni graf prehodov scenarijev enostavno pretvorimo v usmerjenega in dobimo enakovredno nalogo. Pretvorbo naredimo tako, da vse neusmerjene povezave spremenimo v usmerjene s poljubno smerjo. Obratna pretvorba ni tako preprosta, ker v splošnem f_{ij} ni simetrična; če pa f_{ij} je simetrična, potem lahko smer povezav v usmerjenem grafu prehodov scenarijev zanemarimo in tako dobimo neusmerjeni graf.

Celotna prilagodljivost

Na osnovi partikularnih prilagodljivosti lahko definiramo *celotno prilagodljivost*. Pri tem seveda upoštevamo samo tiste partikularne prilagodljivosti f_{ij}, za katere v grafu

G obstaja povezava $(i, j) \in E$. Celotno prilagodljivost bomo označili z

$$f(S_1, S_2, \ldots, S_n),$$

kjer so $S_1, S_2, \ldots, S_n$ partikularne rešitve za scenarije $s_1, s_2, \ldots, s_n$ zaporedoma.

V prilagodljivostnih optimizacijskih problemih gre za iskanje najbolj prilagodljive rešitve, kar dejansko pomeni, da gre za minimizacijo celotne prilagodljivosti. Možni so različni načini izračuna celotne prilagodljivosti, od katerih zopet omenimo predvsem vsoto, maksimum in minimum partikularnih prilagodljivosti.

Torej celotno prilagodljivost lahko definiramo kot vsoto, t.j.

$$f_{sum}(S_1, S_2, \ldots, S_n) = \sum_{(i,j)\in E} f_{ij}(S_i, S_j),$$

čemur bomo v nadaljevanju krajše rekli kar sum prilagodljivost. Če namesto vsote uporabimo maksimum, dobimo

$$f_{max}(S_1, S_2, \ldots, S_n) = \max_{(i,j)\in E} f_{ij}(S_i, S_j),$$

čemur bomo rekli tudi max prilagodljivost, in še zadnji način z minimumom, ki je definiran kot

$$f_{min}(S_1, S_2, \ldots, S_n) = \min_{(i,j)\in E} f_{ij}(S_i, S_j).$$

V problemih prilagajanja gre ponavadi za minimizacijo celotne prilagodljivosti. V tem primeru je smiselna uporaba f_{sum} in f_{max}. (Z uporabo f_{min} dobimo minimizacijo minimuma partikularnih prilagodljivosti, kar nima globljega praktičnega smisla.) Lahko pa namesto minimizacije uporabimo maksimizacijo, ki je smiselna v povezavi z f_{sum} in f_{max}. V primeru uporabe maksimizacije lahko govorimo o *neprilagodljivosti*.

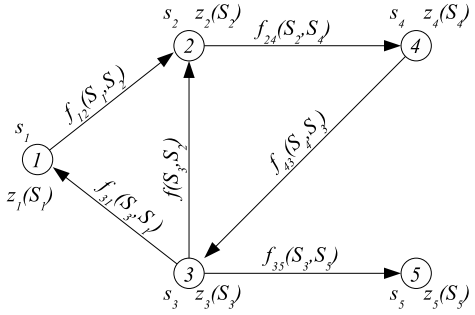
Primer. Podjetje (primer iz predhodnega razdelka) namerava (zaradi tehničnih ali finančnih omejitev) izdelke distribuirati samo med določenimi pari regij. Transport izdelkov med regijami je predstavljen z grafom prehodov scenarijev, v katerem povezava predstavlja pretok izdelkov med dvema regijama. V vsaki od regij podjetje izbere p lokacij za postavitev ponudnikov, partikularna prilagodljivost je cena transporta izdelkov med ponudniki iz dveh različnih povezanih regij. Celotna prilagodljivost je seštevek partikularnih prilagodljivosti; s tem zaobjema celotno ceno distribuiranja izdelkov. Cilj podjetja je seveda to ceno minimizirati.

3.2.4 Prilagodljivostni optimizacijski problem

Sedaj imamo končno na voljo vse potrebno, da sestavimo prilagodljivostni optimizacijski problem P'. Naloga $x' \in I'$ problema P' je torej sestavljena iz enostavnega povezanega grafa prehodov scenarijev $G = (V, E)$ z vozlišči $V = \{1, 2, \ldots, n\}$ in

povezavami E. Vsakemu od vozlišč $i \in V$ je prirejen scenarij $s_i \in I$, kjer je I množica nalog osnovnega optimizacijskega problema P. Poleg tega sta poznana še način izračuna celotne kakovosti $z(S_1, S_2, \ldots, S_n)$ iz partikularnih kakovosti $z_i(S_i)$ in celotne prilagodljivosti $f(S_1, S_2, \ldots, S_n)$ iz partikularnih prilagodljivosti $f_{ij}(S_i, S_j)$. Kriterijska funkcija problema P' vsebuje celotno prilagodljivost. Kriteriji kakovosti pa v problemu P' lahko nastopajo kot del kriterijske funkcije (v tem primeru govorimo o večkriterijski optimizaciji) ali tudi kot omejitve v problemu. Kadar bo govora o prilagodljivosti, bo cilj problema P' minimizacija.

Slika 3.6: Prilagodljivostni optimizacijski problem.



Grafično bomo optimizacijske probleme prilagajanja večkrat prikazali tako kot na sliki 3.6. Prikazan je graf prehodov scenarijev $G = (V, E)$ z vozlišči $V = \{1, 2, 3, 4, 5\}$ in povezavami $E = \{(1, 2), (2, 4), (3, 1), (3, 2), (3, 5), (4, 3)\}$. Vsakemu od vozlišč $i \in V$ je prirejen scenarij s_i in funkcija $z_i(S_i)$ za izračun kakovosti partikularne rešitve S_i za scenarij s_i. Za vsako povezavo $(i, j) \in E$ je podana funkcija $f_{ij}(S_i, S_j)$ za izračun cene prilagoditve S_i v S_j.

3.3 Vrste prilagodljivosti

Omenili smo že več različic prilagodljivosti. Glede na vrsto kriterijske funkcije razlikujemo med sum in max prilagodljivostjo. V primeru maksimizacije lahko vpeljemo še sum in min neprilagodljivost. Govorimo lahko o večkriterijski optimizaciji ali o uporabi kakovosti kot omejitve v problemu.

3.3.1 Problemi prilagodljivih atributov

V naslednjem poglavju bomo pozornost najprej posvetili posebni različici prilagodljivostnih optimizacijskih problemov, ki jih bomo poimenovali *problemi prilagodljivih atributov*. V teh problemih se bomo osredotočili izključno na optimizacijo celotne prilagodljivosti, pri čemer bomo zanemarili partikularno kakovost. Poleg tega bomo podali natančen opis scenarijev, poudarek bo na njihovem preprostem in splošnem opisu. Na ta način bomo dobili preprostejšo in konkretno različico prilagodljivosti, ki je teoretično bolj obvladljiva.

3.3.2 Splošni problemi prilagajanja

Poglavju o problemih prilagodljivih atributov bo sledilo poglavje v katerem bomo v nekatere optimizacijske probleme vpeljali prilagodljivost in na ta način dobili njihove prilagodljivostne različice. Pri obvladovanju prilagodljivostnih različic bomo večkrat uporabili rezultate problemov prilagodljivih atributov.

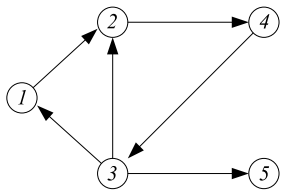
3.3.3 Grafi prehodov scenarijev

Problem prilagajanja je definiran na splošnem grafu prehodov scenarijev. Pri nadaljnji obravnavi se bomo večkrat omejili na določeno vrsto grafa. S tem pogosto poenostavimo obravnavo problema, posledično to vodi v konstrukcijo učinkovitejšega algoritma. Nekaj zanimivejših primerov grafov prehodov scenarijev navajamo v tabelah 3.7 in 3.8. V vseh primerih gre za graf s petimi vozlišči.

Tabela 3.7: Grafi prehodov scenarijev.

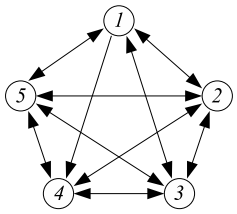
Splošni graf

Splošni graf prehodov scenarijev predstavlja poljuben način prehajanja scenarijev. Reševanje problema prilagajanja na takšnem grafu je pogosto zahtevnejše v primerjavi z ostalimi vrstami grafov, vendar pa je takšen opis najbolj splošen in zato v praksi tudi najbolj uporaben.



Polni graf

Polni graf vsebuje vse možne prehode med danimi scenariji. Gre torej za iskanje takšnih rešitev za posamezne scenarije, ki so si med seboj vse podobne. Nekateri praktični problemi zahtevajo prav to.



Prazen graf

Prazen graf – nasprotje polnega grafa – ne vsebuje nobene povezave. Podobnost posameznih rešitev se v takšnem grafu popolnoma zanemari, vsak scenarij torej predstavlja samostojen optimizacijski problem. S tem pravzaprav zanemarimo bistvo prilagodljivosti, zato se s takšnim grafom prehodov scenarijev v nadaljevanju ne bomo ukvarjali.

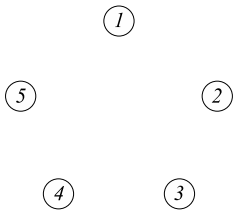
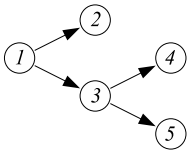


Tabela 3.8: Grafi prehodov scenarijev (2. del).

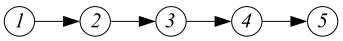
Drevo

Drevo je povezan graf, v katerem velja $|E| = |V| - 1$. Drevo je zelo uporabno za modeliranje praktičnih problemov, poleg tega pa se pogosto zgodi, da je optimizacijski problem precej lažje reševati na drevesu kot na splošnem grafu.



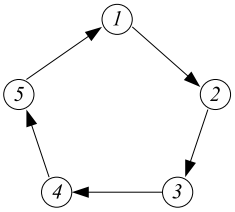
Veriga

Veriga je posebno drevo, v katerem si povezave zaporedno sledijo. Uporabna je za opis problemov, v katerih so prehodi scenarijev zaporedni, t.j. za modeliranje linearnega sosledja dogodkov.



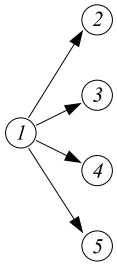
Cikel oz. ciklični graf

Cikel je graf z enakim številom vozlišč in povezav, v katerem je stopnja vsakega vozlišča dva. Cikel je uporaben za modeliranje ponavljajočih se procesov. Včasih ciklu pravimo tudi zaprta veriga.



2-stopenjski graf

2-stopenjski graf prehodov scenarijev je poseben primer drevesa, v katerem iz enega vozlišča izhajajo povezave v vsa ostala. Takšnemu grafu pogosto pravimo tudi zvezda. Ker ga bomo uporabljali za modeliranje problemov, v katerih nastopa en scenarij v sedanjosti, iz njega pa je možnih več povezav v scenarije v prihodnosti, mu raje rečemo 2-stopenjski graf.



Poglavje 4

Problemi prilagodljivih atributov

Čim več povezav spoznamo, tem več vemo o predmetu raziskave.
– John Dewey

Glavni predmet tega poglavja so problemi prilagodljivih atributov. V njih se osredotočimo na optimizacijo prilagodljivosti in na enostaven opis scenarijev. Scenarije opišemo z množico atributov. Najprej predstavimo enotno notacijo in opišemo način izračuna partikularne prilagodljivosti, nato glede na sum in max prilagodljivost definiramo najsplošnejše različice problemov prilagodljivih atributov, za katere obdelamo tudi zahtevnost reševanja, t.j. dokažemo $\mathcal{NP}$-težkost in opišemo njihovo absolutno in relativno aproksimabilnost.

Zatem se lotimo različic, v katerih je potrebno za vsak scenarij izbrati le en atribut. Opišemo njihovo zahtevnost reševanja, nato za te različice opišemo matematične, t.j. kvadratne in linearne celoštevilčne programe. Predstavimo še polinomska natančna algoritma za primer, kadar je graf prehodov scenarijev drevo, in superpolinomska natančna algoritma za splošni graf prehodov scenarijev.

Nato se lotimo dvostopenjskih problemov investiranja, v katerih gre za probleme prilagodljivih atributov z dvostopenjskim grafom prehodov scenarijev. Predstavimo nekatere rezultate o zahtevnosti reševanja, med drugim tudi polinomske natančne in polinomske aproksimacijske algoritme. Opišemo tudi polinomski natančni algoritem za sum prilagodljivost atributov, v katerih je graf prehodov scenarijev veriga. V zadnjem razdelku prikažemo še nekaj potencialnih aplikacij problemov prilagodljivih atributov.

4.1 Definicije problemov

V predhodnem poglavju smo prilagodljivost v optimizacijskih problemih skušali zaobjeti kar se da široko. Povedali smo, da je celotna rešitev problema prilagajanja sestavljena iz (več) partikularnih rešitev. Predstavili smo dva različna kriterija za vrednotenje kakovosti rešitve. Pri prvem gre za vrednotenje kakovosti partikularnih

rešitev (glede na ustrezen scenarij) neodvisno od ostalih partikularnih rešitev, pri drugem pa gre za izračun cene medsebojnega prilagajanja partikularnih rešitev.

Če v danemu problemu prilagodljivosti na scenarij in njegovo partikularno rešitev pogledamo z vidika samostojnega optimizacijskega problema, pri čemer seveda popolnoma izključimo kriterij medsebojnega prilagajanja partikularnih rešitev, potem gre pri problemu pravzaprav za klasično optimizacijo, ki je že dodobra obdelana v literaturi. Poleg tega na ta način popolnoma izključimo samo srž prilagodljivosti, ki jo želimo raziskovati.

To je razlog, da se bomo v tem poglavju osredotočili le na drugi kriterij, t.j. na optimizacijo prilagodljivosti partikularnih rešitev. S tem bomo pozornost našega študija usmerili le na prilagodljivost. Kakovost partikularnih rešitev bomo pri tem izključili, saj bi ta lahko pri študiju prilagodljivosti predstavljala morebitno nepotrebno oviro. V tem poglavju torej želimo prilagodljivost raziskovati v kar najpreprostejši obliki.

Za enostaven in pregleden študij prilagodljivosti je pomemben tudi enoten in splošen način predstavitve scenarijev. V ta namen bomo posamezen scenarij v nadaljevanju predstavili z množico, elemente te množice pa bomo poimenovali *atributi*. Gre torej za imena (in ne vrednosti) spremenljivk. Skupno ime za probleme, ki jih bomo obdelovali v tem poglavju, je *problemi prilagodljivih atributov*.

Atribut lahko (v praktičnem problemu) predstavlja nek objekt, dejanje, akcijo, odločitev ali lastnost itd., kot je npr. premik robotske roke po določeni trajektoriji, cena končnega izdelka, dobavitelj reprodukcijskega materiala, lokacija za postavitev ponudnika itd. Pomen oz. dejanska vsebinska predstavitev atributov za nas z vidika raziskovanja prilagodljivosti ni pomembna, zato bomo v nadaljevanju uporabljali le abstrakten pojem atribut. Vseeno bomo v zadnjem razdelku tega poglavja predstavili nekaj aplikacij problemov prilagodljivih atributov.

Z osredotočenjem na optimizacijo prilagodljivosti in s preprostim ter enovitim načinom opisa scenarijev torej skušamo doseči naslednje:

- da se naša raziskava osredotoča na prilagodljivost; in
- da je naša raziskava prilagodljivosti enostavna in pregledna.

V predhodnem poglavju smo predstavili dve različici optimizacije prilagodljivosti, t.j. sum in max prilagodljivost. Glede na ti dve različici bomo v nadaljevanju obdelali naslednja dva problema:

- Najmanjša $\sum$-Prilagodljivost p-Atributov; in
- Najmanjša max-Prilagodljivost p-Atributov.

Vrednost p je vhodni parameter problemov in predstavlja moč partikularnih dopustnih rešitev, t.j. število atributov, ki jih je potrebno izbrati v vsakem od scenarijev. Kadar je vrednost p poljubna, bomo takšnemu načinu izbiranja atributov rekli tudi prilagodljivost p atributov.

Poleg tega bomo posebej obdelali še preprostejši različici zgornjih dveh problemov, v katerih velja $p = 1$. To sta:

- Najmanjša $\sum$-Prilagodljivost 1-Atributov; in
- Najmanjša max-Prilagodljivost 1-Atributov.

Pri $p = 1$ gre za izbiro natanko enega atributa v vsakem od scenarijev. Včasih bomo tej vrsti rekli prilagodljivost $p = 1$ atributov. Kljub vsem omenjenim poenostavitvam so, kot bomo pokazali v nadaljevanju, vsi v tem podrazdelku našteti problemi prilagodljivosti $\mathcal{NP}$-težki.

4.1.1 Notacija

Za vse omenjene probleme prilagodljivih atributov bomo najprej vpeljali enotno notacijo. Naloge problemov prilagodljivih atributov bomo označevali s četverko (G, I, C, p), pri čemer je

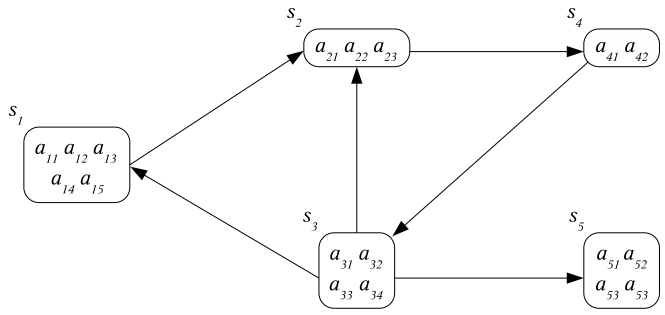
- $G = (V, E)$ graf prehodov scenarijev,
- kjer je $V = \{1, 2, \ldots, n\}$ množica vozlišč in E množica povezav,
- $I = \{s_1, s_2, \ldots, s_n\}$ je množica vozliščem pripadajočih scenarijev,
- kjer vozlišču $i \in V$ pripada scenarij $s_i \in I$,
- ki je opisan z množico atributov $A_i = \{a_{i1}, a_{i2}, \ldots, a_{i|A_i|}\}$,
- $C = \{c_{ij} | (i, j) \in E\}$ je množica funkcij,
- kjer $c_{ij} : A_i \times A_j \to \mathbb{Z}^+$ in
- $c_{ij}(a_{ik}, a_{jl})$ predstavlja ceno prilagoditve $a_{ik} \in A_i$ v $a_{jl} \in A_j$,
- $p \leq \min_{i \in V} |A_i|$ je pozitivno celo število,
- ki določa moč partikularnih rešitev $S_i \subseteq A_i$, kjer $|S_i| = p$.

V nadaljevanju bomo večkrat uporabljali tudi naslednji dve oznaki. Prva $q = \min_{i \in V} |A_i|$ označuje najmanjše število atributov poljubnega scenarija in druga $r = \max_{i \in V} |A_i|$ označuje največje število atributov.

Za naloge problemov Najmanjša $\sum$-Prilagodljivost 1-Atributov in Najmanjša max-Prilagodljivost 1-Atributov vedno velja $p = 1$, zato jih bomo označevali s trojico $(G, I, C) = (G, I, C, 1)$. Oglejmo si naslednji primer naloge prilagodljivih atributov.

Primer. Naj bo (G, I, C) naloga prilagodljivih atributov, kjer je $G = (V, E)$ graf prehodov scenarijev z vozlišči $V = \{1, 2, 3, 4, 5\}$ in povezavami E, kot je prikazano na sliki 4.1. Vozliščem pripadajoči scenariji $s_1, s_2, \ldots, s_5$ so opisani z množicami atributov $A_1 = \{a_{11}, a_{12}, a_{13}, a_{14}, a_{15}\}$, $A_2 = \{a_{21}, a_{22}, a_{23}\}$, $A_3 = \{a_{31}, a_{32}, a_{33}, a_{34}\}$, $A_4 = \{a_{41}, a_{42}\}$, $A_5 = \{a_{51}, a_{52}, a_{53}, a_{54}\}$. Ker scenariji enolično pripadajo vozliščem, so oznake vozlišč na sliki izpuščene, zapisane pa so oznake scenarijev. Tako je poleg vozlišča i zapisana le oznaka scenarija s_i. Poleg tega bomo atribute A_i, ki pripadajo scenariju s_i, pisali znotraj vozlišča i.

Slika 4.1: Primer problema prilagodljivih atributov.



Na ta način enostavno grafično ponazorimo nalogo prilagodljivih atributov. Seveda pri opisu naloge ne smemo pozabiti definirati še cen prilagoditev atributov. V tem primeru definirajmo ceno prilagoditve $c_{ij}(a_{ik}, a_{jl})$ poljubnih dveh atributov $a_{ik} \in A_i$ in $a_{jl} \in A_j$, kjer $(i, j) \in E$, kot $c_{ij}(a_{ik}, a_{jl}) = |i - j| + |k - l|$.

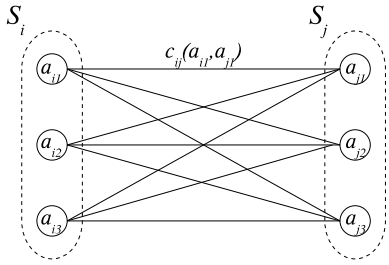
4.1.2 Izračun partikularne prilagodljivosti

Preden se dokončno lotimo natančne definicije vsakega izmed zgoraj naštetih problemov, povejmo še, kako za $(i, j) \in E$ izračunamo partikularno prilagodljivost $f_{ij}(S_i, S_j)$, kjer sta $S_i \subseteq A_i$ in $S_j \subseteq A_j$ partikularni rešitvi z močjo p. Vhodni podatki v nalogi problema prilagajanja atributov so med drugim tudi funkcije $c_{ij}(a_{ik}, a_{jl})$ za vsak $(i, j) \in E$, s katerimi podamo ceno prilagajanja poljubnih dveh atributov $a_{ik} \in A_i$ in $a_{jl} \in A_j$. S pomočjo funkcije c_{ij} izračunamo f_{ij}. Možnih je seveda več različnih načinov izračuna f_{ij}, med katerimi pa nas bo zanimal najcenejši način, t.j. zanimala nas bo cena prilagajanja S_i v S_j, pri katerem je vrednost f_{ij} najmanjša. Pokazali bomo, kako lahko to enostavno izračunamo s pomočjo problema Najmanjša Dodelitev, ki smo ga opisali v podrazdelku 2.5.2.

Naj bosta $S_i = \{a_{i1}, a_{i2}, \ldots, a_{ip}\}$ in $S_j = \{a_{j1}, a_{j2}, \ldots, a_{jp}\}$ partikularni rešitvi scenarijev s_i oz. s_j. Atribute, ki sestavljajo S_i in S_j, pretvorimo v vozlišča in sestavimo polni dvodelni graf $H_{ij} = (S_i \cup S_j, F)$, kjer $F = S_i \times S_j$, kot je prikazano na sliki 4.2 (za primer $p = 3$). Cena povezave v tako zgrajenem grafu je enaka ceni pretvorbe ustreznih dveh atributov. Natančneje, cena povezave $(a_{ik}, a_{jl}) \in F$ je enaka $c_{ij}(a_{ik}, a_{jl})$. Dvodelni graf H_{ij} sestavlja $2p$ vozlišč – levi in desni del vsak

posebej vsebujeta p vozlišč – zato v H_{ij} vedno obstaja popolno ujemanje velikosti p.

Slika 4.2: Izračun partikularne prilagodljivosti.



Naj bo M_{ij} rešitev problema Najmanjša Dodelitev na grafu H_{ij}. Velja $|M_{ij}| = p$. Povezave v M_{ij} predstavljajo izbrane pretvorbe atributov med S_i in S_j, in sicer $(a_{ik}, a_{jl}) \in M_{ij}$ pomeni, da pretvorimo atribut a_{ik} v a_{jl}. Ker je $|M_{ij}| = p$, so s tem zaobjeti vsi atributi iz S_i in S_j. Partikularna prilagodljivost f_{ij} je enaka ceni ujemanja M_{ij}. Torej

$$f_{ij}(S_i, S_j) = \sum_{(a_{ik}, a_{jl}) \in M_{ij}} c_{ij}(a_{ij}, a_{jl}).$$

Ker je cena ujemanja M_{ij} najmanjša, je prav gotovo tudi f_{ij} najmanjša.

Opisan izračun je uporaben tako v problemih sum prilagodljivosti kot v problemih max prilagodljivosti. Morda dobimo vtis, da je računanje partikularne prilagodljivosti na takšen način, kjer je potrebno za vsak problem prilagodljivosti rešiti $|E|$ problemov Najmanjša Dodelitev, precej zapleteno. V nadaljevanju bomo videli, da ni vedno tako.

Izbira enega atributa

Bolj preprost je izračun f_{ij} v primeru $p = 1$, kjer za vsak scenarij izberemo natanko en atribut. V tem primeru velja preprosto

$$f_{ij}(S_i, S_j) = c_{ij}(a_i, a_j),$$

kjer $S_i = \{a_i\}$ in $S_j = \{a_j\}$.

4.1.3 Prilagodljivost p atributov

Spoznali smo torej način, kako izračunati najcenejšo ceno medsebojne prilagoditve dveh partikularnih rešitev S_i in S_j. Sedaj se lahko lotimo definicij problemov prilagodljivosti. Najprej definirajmo splošne probleme, v katerih v vsakem scenariju izberemo p atributov. Dopustne rešitve teh problemov so zaporedja $(S_1, S_2, \ldots, S_n)$ množic atributov $S_i \subseteq A_i$, kjer $|S_i| = p$ za vsak $1 \leq i \leq n$. Množica S_i vsebuje

atribute, izbrane v scenariju s_i. Posebej bomo definirali sum in max prilagodljivost atributov. Najprej definirajmo prvo.

Problem 4.6: NAJMANJŠA $\sum$-PRILAGODLJIVOST p-ATRIBUTOV

Naloga: Četvorka (G, I, C, p), kjer je $G = (V, E)$ graf prehodov scenarijev. Vozlišču $i \in V$ pripada scenarij $s_i \in I$, ki je opisan z atributi A_i. Za vsak par atributov $a_{ik} \in A_i$ in $a_{jl} \in A_j$, kjer $(i, j) \in E$, je $c_{ij}(a_{ik}, a_{jl}) \in \mathbb{Z}^+$ cena prilagajanja atributov. Parameter $p \in \mathbb{Z}^+$, da za vsak i velja $p \leq |A_i|$.
Dopustna rešitev: Zaporedje $(S_1, S_2, \ldots, S_n)$ množic $S_i \subseteq A_i$, kjer $|S_i| = p$ za vsak $1 \leq i \leq n$.
Kakovost dopustne rešitve: $f_{sum}(S_1, S_2, \ldots, S_n) = \sum_{(i,j)\in E} f_{ij}(S_i, S_j)$.
Cilj: Minimizacija.

Problem NAJMANJŠA $\sum$-PRILAGODLJIVOST p-ATRIBUTOV je v vsakem scenariju izbrati p atributov, tako da je vsota partikularnih prilagodljivosti f_{ij} najmanjša. Drugi problem se nanaša na max prilagodljivost, pri kateri je celotna cena prilagajanja enaka maksimumu partikularnih prilagodljivosti, ali povedano drugače, da je najdražje partikularno prilagajanje najcenejše.

Problem 4.7: NAJMANJŠA max-PRILAGODLJIVOST p-ATRIBUTOV

Naloga: Četvorka (G, I, C, p), kjer je $G = (V, E)$ graf prehodov scenarijev. Vozlišču $i \in V$ pripada scenarij $s_i \in I$, ki je opisan z atributi A_i. Za vsak par atributov $a_{ik} \in A_i$ in $a_{jl} \in A_j$, kjer $(i, j) \in E$, je $c_{ij}(a_{ik}, a_{jl}) \in \mathbb{Z}^+$ cena prilagajanja atributov. Parameter $p \in \mathbb{Z}^+$, da za vsak i velja $p \leq |A_i|$.
Dopustna rešitev: Zaporedje $(S_1, S_2, \ldots, S_n)$ množic $S_i \subseteq A_i$, kjer $|S_i| = p$, za vsak $1 \leq i \leq n$.
Kakovost dopustne rešitve: $f_{max}(S_1, S_2, \ldots, S_n) = \max_{(i,j)\in E} f_{ij}(S_i, S_j)$.
Cilj: Minimizacija.

V definiciji obeh problemov je cena c_{ij} prilagajanj atributov nenegativno celo število. Probleme, v katerih so vsi c_{ij} pozitivna racionalna števila, pretvorimo na zgornji opis preprosto tako, da vse c_{ij} množimo z dovolj veliko konstanto. Podobno lahko probleme, v katerih je c_{ij} lahko negativno število, pretvorimo na zgornji opis tako, da vsem c_{ij} prištejemo dovolj veliko konstanto.

4.1.4 Prilagodljivost $p = 1$ atributov

V nadaljevanju bomo večkrat obdelovali probleme, v katerih v vsakem scenariju izberemo natanko en atribut, zato jih na tem mestu posebej definirajmo. Ker gre za izbiro samo enega atributa, bomo dopustne rešitve pisali kot zaporedja $(a_1, a_2, \ldots, a_n)$, kjer $a_i \in A_i$ predstavlja atribut, izbran v scenariju s_i. Velja seveda $S_i = \{a_i\}$. Zopet najprej definirajmo problem sum prilagodljivosti.

Problem 4.8: NAJMANJŠA $\sum$-PRILAGODLJIVOST 1-ATRIBUTOV

Naloga: Trojica (G, I, C), kjer je $G = (V, E)$ graf prehodov scenarijev. Vozlišču $i \in V$ pripada scenarij $s_i \in I$, ki je opisan z atributi A_i. Za vsak par atributov $a_{ik} \in A_i$ in $a_{jl} \in A_j$, kjer $(i, j) \in E$, je $c_{ij}(a_{ik}, a_{jl}) \in \mathbb{Z}^+$ cena prilagajanja atributov.
Dopustna rešitev: Zaporedje $(a_1, a_2, \ldots, a_n)$ atributov $a_i \in A_i$.
Kakovost dopustne rešitve: $f_{sum}(a_1, a_2, \ldots, a_n) = \sum_{(i,j)\in E} c_{ij}(a_i, a_j)$.
Cilj: Minimizacija.

Problem NAJMANJŠA $\sum$-PRILAGODLJIVOST 1-ATRIBUTOV je za vsak scenarij izbrati natanko en atribut, tako da je vsota cen vseh prilagajanj atributov najmanjša. Druga različica problema prilagodljivih atributov se nanaša na max prilagodljivost. Formalna definicija problema je naslednja.

Problem 4.9: NAJMANJŠA max-PRILAGODLJIVOST 1-ATRIBUTOV

Naloga: Trojica (G, I, C), kjer je $G = (V, E)$ graf prehodov scenarijev. Vozlišču $i \in V$ pripada scenarij $s_i \in I$, ki je opisan z atributi A_i. Za vsak par atributov $a_{ik} \in A_i$ in $a_{jl} \in A_j$, kjer $(i, j) \in E$, je $c_{ij}(a_{ik}, a_{jl}) \in \mathbb{Z}^+$ cena prilagajanja atributov.
Dopustna rešitev: Zaporedje $(a_1, a_2, \ldots, a_n)$ atributov $a_i \in A_i$.
Kakovost dopustne rešitve: $f_{max}(a_1, a_2, \ldots, a_n) = \max_{(i,j)\in E} c_{ij}(a_i, a_j)$.
Cilj: Minimizacija.

Problem NAJMANJŠA max-PRILAGODLJIVOST 1-ATRIBUTOV je torej za vsak scenarij izbrati natanko en atribut, tako da je cena najdražjega prilagajanja dveh atributov najmanjša.

4.2 Zahtevnost reševanja

4.2.1 Velikost prostora rešitev

Oglejmo si velikost prostora rešitev problemov prilagodljivih atributov. Splošni različici s poljubnim p sta NAJMANJŠA $\sum$-PRILAGODLJIVOST p-ATRIBUTOV in NAJMANJŠA max-PRILAGODLJIVOST p-ATRIBUTOV. Dopustne rešitve obeh problemov so zaporedja $(S_1, S_2, \ldots, S_n)$, kjer $S_i \subseteq A_i$ in $|S_i| = p$. Za vsakega od s_i lahko p atributov izberemo na $\binom{|A_i|}{p}$ možnih načinov. Velikost prostora rešitev je torej omejena[1] med

$$\binom{q}{p}^n \leq \prod_{i\in V} \binom{|A_i|}{p} \leq \binom{r}{p}^n. \tag{4.1}$$

[1]Spomnimo, da $q = \min_{i\in V} |A_i|$ in $r = \max_{i\in V} |A_i|$.

Velja seveda

$$\binom{q}{p}^n = \Omega(\left(\frac{q}{p}\right)^{pn}).$$

Velikost prostora rešitev je torej eksponentno odvisna od p in n ter polinomsko odvisna od q/p. (Velja $q \geq p$.) Predpostavimo, da sta p in n nespremenljiva, torej konstantna in ne parametra problema. V tem primeru je v polinomskem času mogoče temeljito preiskovanje prostora rešitev. Vendar čas takšnega reševanja (kljub polinomski odvisnosti) že pri majhnih p in n hitro preraste vse razumne meje, zato se temeljito preiskovanje v praksi ne obnese.

V najslabšem primeru, to je pri maksimumu binomskega simbola, je $p = q/2$. Vstavimo to v enačbo (4.1) in ugotovimo, da je število dopustnih rešitev omejeno z $\Omega(2^{qn/2})$, torej je eksponentno odvisno.

Oglejmo si še število dopustnih rešitev za prilagodljivost $p = 1$ atributov. Za problema Najmanjša $\sum$-Prilagodljivost 1-Atributov in Najmanjša max-Prilagodljivost 1-Atributov je število dopustnih rešitev omejeno

$$q^n \leq \prod_{i \in V} |A_i| \leq r^n.$$

Torej velja, da je število dopustnih rešitev za $p = 1$ enako $\Omega(q^n)$ oz. $O(r^n)$. To je polinomsko odvisno od q oz. r in eksponentno odvisno od števila scenarijev n.

Probleme v katerih je število scenarijev n zelo majhno, morda lahko poskusimo optimalno reševati s temeljitim preiskovanjem prostora rešitev. V nadaljevanju bomo pokazali, da so vsi omenjeni problemi $\mathcal{NP}$-težki.

4.2.2 $\mathcal{NP}$-težkost prilagodljivosti p atributov

Za dokazovanje pripadnosti problemov prilagodljivih atributov razredu $\mathcal{NP}$-težkih problemov bomo uporabili postopek prevedb, opisan v podrazdelku 2.2.3. Naj bo $P \in \mathcal{NPO}$ problem, za katerega želimo pokazati $\mathcal{NP}$-težkost. Najprej pokažemo $\mathcal{NP}$-polnost odločitvene različice P_{odl}, nato s pomočjo trditve 2.3 zapišemo posledico o $\mathcal{NP}$-težkosti P.

Sum prilagodljivost

Najprej bomo dokazali, da je Najmanjša $\sum$-Prilagodljivost p-Atributov $\mathcal{NP}$-težek problem. V ta namen definirajmo njegovo odločitveno različico.

Problem 4.10: $\sum$-Prilagodljivost p-Atributov

Naloga: Naloga (G, I, C, p) prilagodljivih atributov in parameter $B \in \mathbb{Z}^+$.
Vprašanje: Ali obstaja zaporedje $(S_1, S_2, \ldots, S_n)$, kjer $S_i \subseteq A_i$ in $|S_i| = p$, da $\sum_{(i,j) \in E} f_{ij}(S_i, S_j) \leq B$?

Vprašanje problema $\sum$-Prilagodljivost p-Atributov je torej, ali obstaja takšna rešitev $(S_1, S_2, \ldots, S_n)$, da njena cena ni večja od B. Odločitvena različica torej ne povprašuje po konstrukciji takšne rešitve, ampak samo po njenem obstoju.

V prevedbi bomo potrebovali nek $\mathcal{NP}$-poln problem. V ta namen definirajmo Pokritje Množice, ki je dobro znan $\mathcal{NP}$-poln problem [GJ79, CLRS01].

Problem 4.11: Pokritje Množice

Naloga: Množica $U = \{u_1, \ldots, u_n\}$ in družina $\mathcal{U} = \{U_1, U_2, \ldots, U_m\}$ podmnožic $U_i \subseteq U$, kjer velja $\cup_{X \in \mathcal{U}} X = U$, ter parameter $p \in \mathbb{N}$.
Vprašanje: Ali obstaja poddružina $T \subseteq \mathcal{U}$, kjer $|T| \leq p$, da $\cup_{X \in T} X = U$?

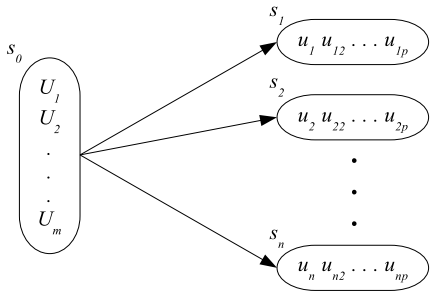
Problem Pokritje Množice je torej ugotoviti, ali obstaja družina T kvečjemu p podmnožic $U_i \subseteq \mathcal{U}$, da $\cup_{X \in T} X = U$, čemur pravimo, da s T *pokrijemo* U. Zapišimo in dokažimo naslednjo trditev.

Trditev 4.8 ▶ $\sum$-Prilagodljivost p-Atributov *je* $\mathcal{NP}$*-poln problem.*

Dokaz. Enostavno lahko pokažemo, da je $\sum$-Prilagodljivost p-Atributov v $\mathcal{NP}$. Za podano (G, I, C, p) uganemo rešitev in preverimo, če njena cena ni večja od B. Opišimo prevedbo.

Iz naloge problema Pokritje Množice, t.j. iz množice U, družine $\mathcal{U}$ in parametra p, bomo zasnovali nalogo (G, I, C, p) za $\sum$-Prilagodljivost p-Atributov. Konstrukcija naloge je prikazana na sliki 4.3.

Slika 4.3: Konstrukcija naloge v prevedbi.



Ustvarjeni graf G je sestavljen iz $n + 1$ vozlišč, katerim pripadajo scenariji $s_0, s_1, \ldots, s_n$. V G dodamo tudi n povezav iz vozlišča 0 v vsa ostala vozlišča. Natančneje, za vsak i, kjer $1 \leq i \leq n$, v G nastopa povezava $(0, i) \in E$. Scenarij s_0 je opisan z množico $A_0 = \mathcal{U}$. Izbira p atributov v A_0 predstavlja izbiro p množic v $\mathcal{U}$. Vsak od ostalih scenarijev je opisan z natanko enim od elementov iz U in $p - 1$ pomožnimi atributi. Natančneje, scenarij s_i, kjer $1 \leq i \leq n$, je opisan z množico $A_i = \{u_i, u_{i2}, u_{i3}, \ldots, u_{ip}\}$, kjer $u_i \in U$. Za pomožne atribute $u_{il} \in A_i$, kjer $2 \leq l \leq p$, velja $u_{il} \notin U$.

V G smo ustvarili $n+1$ vozlišč oz. scenarijev, n povezav in $m+np$ atributov. Opisana prevedba je torej prav gotovo polinomska. Definirajmo še cene c_{0j} prilagoditev atributov, kjer $(0, j) \in E$. Naj bo $U_i \in A_0$. Za atribut $u_j \in A_j$ definirajmo

$$c_{0j}(U_i, u_j) = \begin{cases} 0 & u_j \in U_i \\ \infty & \text{sicer.} \end{cases}$$

Za vse ostale (t.j. pomožne) atribute $u_{jl} \in A_j$, kjer $2 \leq l \leq p$, pa definirajmo

$$c_{0j}(U_i, u_{jl}) = 0.$$

Polinomska prevedba je tako popolnoma definirana, preostane nam samo še, da preverimo njeno pravilnost. To naredimo tako, da pokažemo veljavnost naslednje trditve. Za poljubno nalogo problema Pokritje Množice obstaja rešitev T z največ p elementi natanko takrat, ko je $S = (T, A_1, A_2, \ldots, A_n)$ rešitev s ceno 0 za ustrezno nalogo problema $\sum$-Prilagodljivost p-Atributov.

($\Rightarrow$) Naj bo T, kjer $|T| = p$, rešitev problema Pokritje Množice. (Če $|T| < p$, v T dodamo $p - |T|$ podmnožic $U_i \in \mathcal{U}$, ki še niso v T.) Rešitev $\sum$-Prilagodljivost p-Atributov je oblike $S = (T, A_1, A_2, \ldots, A_n)$, kar pomeni, da v s_0 izberemo atribute T, v vseh ostalih scenarijih s_j, kjer $1 \leq j \leq n$, pa izberemo kar vse atribute A_j. Očitno je, da je S dopustna rešitev $\sum$-Prilagodljivost p-Atributov.

Izračunajmo še ceno S, t.j. $f_{sum}(S) = \sum_{j=1}^{n} f_{0j}(T, A_j)$. Najprej si oglejmo $f_{0j}(T, A_j)$. Po definiciji c_{0j} za pomožne atribute $u_{jl} \in A_j$, kjer $2 \leq l \leq p$, za vse $U_i \in \mathcal{U}$ velja $c_{0j}(U_i, u_{jl}) = 0$. Izbira pomožnih atributov u_{jl} torej ne spreminja vrednosti f_{0j}, ki je potemtakem odvisna le od T in $u_j \in A_j$. Za $1 \leq j \leq n$ torej velja $f_{0j}(T, A_j) = \min_{U_i \in T} c_{0j}(U_i, u_j)$. Ker je T pokritje U, za vse $u_j \in U$ obstaja $U_i \in T$, da velja $u_j \in U_i$. Za tak U_i je $c_{0j}(U_i, u_j) = 0$. Sledi $f_{0j}(T, A_j) = 0$ za vse $1 \leq j \leq n$. Torej $f_{sum}(S) = 0$.

($\Leftarrow$) Naj bo $S = (T, A_1, A_2, \ldots, A_n)$ rešitev s ceno 0 za $\sum$-Prilagodljivost p-Atributov. Torej velja $f_{0j}(T, A_j) = 0$ za $1 \leq j \leq n$. Zopet pomožnih atributov ni potrebno upoštevati. Za $1 \leq j \leq n$ velja torej $0 = f_{0j}(T, A_j) = \min_{U_i \in T} c_{0j}(U_i, u_j)$ in nadalje obstaja $U_i \in T$, da $c_{0j}(U_i, u_j) = 0$, iz česar po definiciji c_{0j} sledi $u_j \in U_i$. Potemtakem je T pokritje U. Velja $|T| = p$. □

Predmet raziskovanja v tem delu so optimizacijski problemi, zato je za nas pravzaprav bolj zanimiva naslednja posledica pravkar dokazane trditve.

Posledica 4.9 ▶ Najmanjša $\sum$-Prilagodljivost p-Atributov *je $\mathcal{NP}$-težek optimizacijski problem.*

Max prilagodljivost

Zelo podobno kot za problem NAJMANJŠA $\sum$-PRILAGODLJIVOST p-ATRIBUTOV lahko pokažemo tudi $\mathcal{NP}$-težkost problema NAJMANJŠA max-PRILAGODLJIVOST p-ATRIBUTOV. Zapišimo odločitveno različico problema.

Problem 4.12: max-PRILAGODLJIVOST p-ATRIBUTOV

Naloga: Naloga (G, I, C, p) prilagodljivih atributov in parameter $B \in \mathbb{Z}^+$.
Vprašanje: Ali obstaja zaporedje $(S_1, S_2, \ldots, S_n)$, kjer $S_i \subseteq A_i$ in $|S_i| = p$, da $\max_{(i,j)\in E} f_{ij}(S_i, S_j) \leq B$?

In naslednjo trditev.

Trditev 4.10 ▶ max-PRILAGODLJIVOST p-ATRIBUTOV *je* $\mathcal{NP}$*-poln problem.*

Dokaz. Prav gotovo tudi max-PRILAGODLJIVOST p-ATRIBUTOV pripada razredu $\mathcal{NP}$. V prevedbi zopet uporabimo POKRITJE MNOŽICE. Konstrukcija naloge (G, I, C, p) problema max-PRILAGODLJIVOST p-ATRIBUTOV iz naloge problema POKRITJE MNOŽICE je enaka konstrukciji iz dokaza trditve 4.8. Pravilnost konstrukcije pokažemo enako kot v omenjenem dokazu, le da tokrat namesto f_{sum} uporabimo f_{max}, kar pa sklepanja bistveno ne spremeni. □

Zapišimo še naslednjo posledico.

Posledica 4.11 ▶ NAJMANJŠA max-PRILAGODLJIVOST p-ATRIBUTOV *je* $\mathcal{NP}$*-težek optimizacijski problem.*

Zahtevnost reševanja na drevesu

Oglejmo si še enkrat graf G, ki smo ga uporabili v dokazih trditev 4.8 in 4.10. Opazimo, da je G drevo. Če pogledamo še natančneje, gre pravzaprav za zvezdo oz. za dvostopenjski graf prehodov scenarijev (glej podrazdelek 3.3.3). Ker gre za dve izmed bolj pomembnih oblik grafa prehodov scenarijev, posebej zapišimo še naslednjo ugotovitev.

Posledica 4.12 ▶ *Problema* NAJMANJŠA $\sum$-PRILAGODLJIVOST p-ATRIBUTOV *in* NAJMANJŠA max-PRILAGODLJIVOST p-ATRIBUTOV*, v katerih je graf prehodov scenarijev drevo ali zvezda, sta* $\mathcal{NP}$*-težka optimizacijska problema.*

4.2.3 $\mathcal{NP}$-težkost prilagodljivosti $p = 1$ atributov

Ugotovili smo torej, da sta obe vrsti prilagodljivosti p atributov $\mathcal{NP}$-težka problema; to velja tudi v primeru, ko je graf prehodov scenarijev drevo ali celo zvezda. Vprašajmo se, kako je z zahtevnostjo reševanja, kadar velja $p = 1$, t.j. če želimo v vsakem od scenarijev izbrati natanko en atribut. Podobno kot zgoraj bomo tudi za $p = 1$ pokazali $\mathcal{NP}$-težkost obeh vrst prilagodljivosti na splošnem grafu prehodov scenarijev.

Sum prilagodljivost

Zopet se najprej lotimo sum prilagodljivosti in definirajmo odločitveno različico problema Najmanjša $\sum$-Prilagodljivost 1-Atributov.

Problem 4.13: $\sum$-Prilagodljivost 1-Atributov

Naloga: Naloga (G, I, C) prilagodljivih atributov in parameter $B \in \mathbb{Z}^+$.
Vprašanje: Ali obstaja zaporedje $(a_1, a_2, \ldots, a_n)$ atributov, kjer $a_i \in A_i$, da $\sum_{(i,j)\in E} c_{ij}(a_i, a_j) \leq B$?

Na problem $\sum$-Prilagodljivost 1-Atributov bomo prevedli naslednji problem, ki velja za enega najbolj znanih $\mathcal{NP}$-polnih problemov [GJ79].

Problem 4.14: Trgovski Potnik

Naloga: Množica mest $\{c_1, c_2, \ldots, c_n\}$, matrika $D = (d_{ij})_{n\times n}$, kjer $d_{ij} \in \mathbb{Z}^+$ predstavlja razdaljo od mesta c_i do c_j in parameter $B \in \mathbb{Z}^+$.
Vprašanje: Ali obstaja permutacija $T = (c_{i_1}, c_{i_2}, \ldots, c_{i_n})$ mest, da je njena kakovost $m(T) = \sum_{k=1}^{n-1} d_{i_k,i_{k+1}} + d_{i_n,i_1}$ kvečjemu B, t.j. $m(T) \leq B$?

Trgovski Potnik sprašuje po tem, ali obstaja takšen obhod mest, pri katerem gremo skozi vsako od podanih mest natanko enkrat, da celotna prehojena pot ni večja od B. Trgovski Potnik je $\mathcal{NP}$-poln tudi v nekaterih *lažjih* različicah, kot npr. če razdalje d_{ij} zadoščajo trikotniški neenakosti[2], ali če so d_{ij} razdalje med točkami v evklidski ravnini.

Zapišimo in dokažimo naslednji izrek.

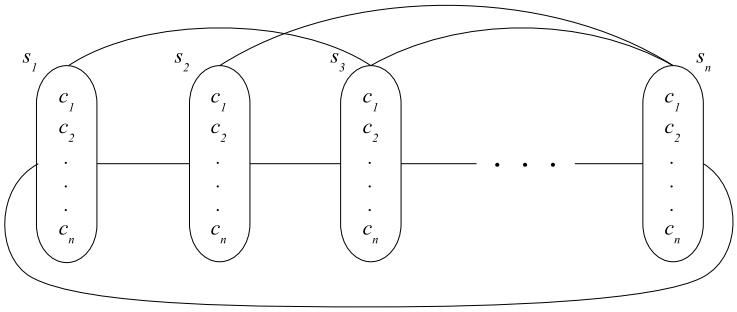
Trditev 4.13 ▶ $\sum$-Prilagodljivost 1-Atributov *je $\mathcal{NP}$-poln problem.*

Dokaz. Prav gotovo $\sum$-Prilagodljivost 1-Atributov pripada razredu $\mathcal{NP}$. Opišimo prevedbo problema Trgovski Potnik. Iz naloge za problem Trgovski Potnik, t.j. množice mest $\{c_1, c_2, \ldots, c_n\}$, matrike D in parametra B, bomo sestavili nalogo za $\sum$-Prilagodljivost 1-Atributov, t.j. graf $G = (V, E)$, množice

[2]Kadar velja $d_{ij} \leq d_{ik} + d_{kj}$ za vse $1 \leq i, j, k \leq n$.

$A_1, A_2, \ldots, A_n$ atributov in cene c_{ij} za vsak $(i, j) \in E$. Konstrukcija je prikazana na sliki 4.4.

Slika 4.4: Konstrukcija grafa prehodov scenarijev.



Ustvarimo poln graf G z n vozlišči oz. scenariji. Atributi naj predstavljajo mesta in vsak od scenarijev naj bo opisan s celotno množico mest. Natančneje, za vsak $i \in V$ definirajmo $A_i = \{c_1, c_2, \ldots, c_n\}$. Izbira atributa (in s tem mesta) v scenariju s_i predstavlja i-to obiskano mesto. Vsako mesto je potrebno obiskati natanko enkrat, kar dosežemo z ustreznim definiranjem cen prilagajanj atributov.

Naj bo $N = \{(n, 0)\} \cup \{(i, i+1) | 1 \leq i < n\}$ množica povezav med "neposrednimi sosedi". Za vsak $(i, j) \in E$ definirajmo ceno $c_{ij}(c_k, c_l)$ prilagoditve atributov $c_k \in A_i$ in $c_l \in A_j$ kot

$$c_{ij}(c_k, c_l) = \begin{cases} \infty & k = l \\ d_{kl} & (i, j) \in N \\ 0 & \text{sicer.} \end{cases}$$

S prevedbo smo ustvarili n vozlišč, $n(n-1)/2$ povezav in n^2 atributov. Polinomska prevedba je tako definirana, potrebno je dokazati še njeno pravilnost. Velja naslednja trditev. Za vsako nalogo problema Trgovski Potnik obstaja rešitev $T = (c_{l_1}, c_{l_2}, \ldots, c_{l_n})$ s ceno $m(T)$ natanko takrat, ko je T rešitev s ceno $f_{sum}(T) = m(T)$ ustrezne naloge problema $\sum$-Prilagodljivost 1-Atributov.

($\Rightarrow$) Naj bo $T = (c_{l_1}, c_{l_2}, \ldots, c_{l_n})$ rešitev s ceno $m(T)$ za nalogo problema Trgovski Potnik. Izračunajmo $f_{sum}(T)$. Ker je T permutacija, za vse $1 \leq i, j \leq n$ in $i \neq j$ velja $l_i \neq l_j$. Po definiciji c_{ij} torej za $(i, j) \in N$ velja $c_{ij}(c_{l_i}, c_{l_j}) = d_{l_i, l_j}$ in za

$(i,j) \in E/N$ velja $c_{ij}(c_{l_i,l_j}) = 0$. Potemtakem lahko zapišemo.

$$\begin{aligned} f_{sum}(T) &= \sum_{(i,j)\in E} c_{ij}(c_{l_i}, c_{l_j}) = \\ &= \sum_{(i,j)\in E/N} c_{ij}(c_{l_i}, c_{l_j}) + \sum_{(i,j)\in N} c_{ij}(c_{l_i}, c_{l_j}) = \\ &= \sum_{(i,j)\in N} c_{ij}(c_{l_i}, c_{l_j}) = \\ &= \sum_{(i,j)\in N} d_{l_i,l_j} = \\ &= \sum_{k=1}^{n-1} d_{l_k,l_{k+1}} + d_{l_n,l_1} = \\ &= m(T). \end{aligned}$$

($\Leftarrow$) Naj bo $T = (c_{l_1}, c_{l_2}, \ldots, c_{l_n})$ rešitev problema $\sum$-Prilagodljivost 1-Atributov s ceno $f_{sum}(T)$. Velja $f_{sum}(T) < \infty$. Ker $f_{sum}(T) \neq \infty$, za vse $(i,j) \in E$ velja $c_{ij}(c_{l_i}, c_{l_j}) \neq \infty$ in s tem tudi $l_i \neq l_j$. Torej je T permutacija, t.j. dopustna rešitev za problem Trgovski Potnik. Nato podobno kot pri pogoju potrebnosti ugotovimo še $f_{sum}(T) = m(T)$. □

Zapišimo še naslednjo posledico.

Posledica 4.14 ▶ Najmanjša $\sum$-Prilagodljivost 1-Atributov *je $\mathcal{NP}$-težek optimizacijski problem.*

Max prilagodljivost

Lotimo se še problema Najmanjša max-Prilagodljivost 1-Atributov. Najprej definirajmo njegovo odločitveno različico.

Problem 4.15: max-Prilagodljivost 1-Atributov

Naloga: Naloga (G, I, C) prilagodljivih atributov in parameter $B \in \mathbb{Z}^+$.
Vprašanje: Ali obstaja zaporedje $(a_1, a_2, \ldots, a_n)$ atributov, kjer $a_i \in A_i$, da $\max_{(i,j)\in E} c_{ij}(a_i, a_j) \leq B$?

Njegovo $\mathcal{NP}$-polnost bomo dokazali tako, da bomo nanj prevedli naslednji dobro znani $\mathcal{NP}$-polni problem [GJ79].

Problem 4.16: Hamiltonov Cikel

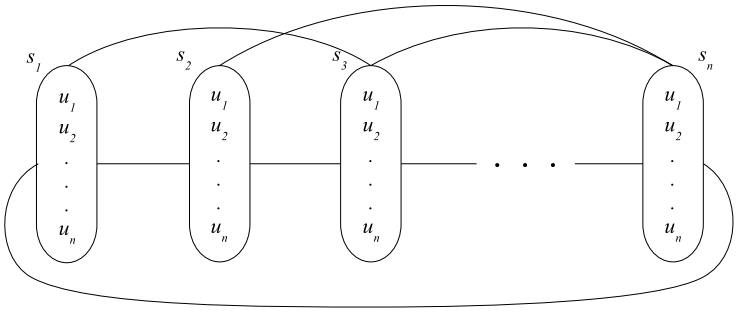
Naloga: Graf $G = (V, E)$.
Vprašanje: Ali obstaja cikel $T = (v_{i_1}, v_{i_2}, \ldots, v_{i_n})$ v grafu G, da vsako vozlišče $v \in V$ obiščemo natanko enkrat?

Zapišimo in dokažimo naslednjo trditev.

Trditev 4.15 ▶ max-Prilagodljivost 1-Atributov *je $\mathcal{NP}$-poln problem.*

Dokaz. Prav gotovo max-Prilagodljivost 1-Atributov pripada razredu $\mathcal{NP}$. Lotimo se konstrukcije. Iz naloge Hamiltonov Cikel, t.j. grafa $H = (U, F)$, sestavimo nalogo za max-Prilagodljivost 1-Atributov, t.j. graf $G = (V, E)$, množico atributov $A_1, A_2, \ldots, A_n$ in cene c_{ij}. Konstrukcija grafa G je pravzaprav enaka kot v dokazu trditve 4.13.

Slika 4.5: Prevedba.



Ustvarimo poln graf $G = (V, E)$ na $n = |U|$ vozliščih. (Glej sliko 4.5.) Za vse $i \in V$ definirajmo $A_i = U = \{u_1, u_2, \ldots, u_n\}$. Izbira atributa iz A_i v s_i predstavlja i-to obiskano vozlišče Hamiltonovega cikla. Potrebno je definirati še cene prilagajanj atributov.

Naj bo $N = \{(n, 0)\} \cup \{(i, i+1) | 1 \leq i < n\}$. Za vsako povezavo $(i, j) \in E$ definirajmo ceno $c_{ij}(u_k, u_l)$ prilagoditve atributov $u_k \in A_i$ in $u_l \in A_j$ kot

$$c_{ij}(u_k, u_l) = \begin{cases} 0 & (i,j) \in N, (u_k, u_l) \in F \\ \infty & (i,j) \in N, (u_k, u_l) \notin F \\ \infty & (i,j) \notin N, k = l \\ 0 & (i,j) \notin N, k \neq l. \end{cases}$$

Pokažimo še, da za vsako nalogo problema Hamiltonov Cikel obstaja rešitev $T = (u_{l_1}, u_{l_2}, \ldots, u_{l_n})$ natanko takrat, ko je T rešitev s ceno 0 za ustrezno nalogo problema max-Prilagodljivost 1-Atributov.

($\Rightarrow$) Naj bo $T = (u_{l_1}, u_{l_2}, \ldots, u_{l_n})$ rešitev problema Hamiltonov Cikel. Izračunajmo $f_{max}(T)$. Ker je T permutacija, za vse $(i, j) \in E/N$ velja $l_i \neq l_j$. Zatorej za vse $(i, j) \in E/N$ velja $c_{ij}(u_{l_i}, u_{l_j}) = 0$. Nadalje, ker je T cikel, za vse $(i, j) \in N$

velja $(u_{l_i}, u_{l_j}) \in F$ in s tem $c_{ij}(u_{l_i}, u_{l_j}) = 0$. Torej lahko zapišemo.

$$\begin{aligned} f_{max}(T) &= \max_{(i,j)\in E} c_{ij}(u_{l_i}, u_{l_j}) = \\ &= \max[\max_{(i,j)\in E/N} c_{ij}(u_{l_i}, u_{l_j}), \max_{(i,j)\in N} c_{ij}(u_{l_i}, u_{l_j}) \ ] = \\ &= \max_{(i,j)\in N} c_{ij}(u_{l_i}, u_{l_j}) = \\ &= 0. \end{aligned}$$

($\Leftarrow$) Naj bo $T = (u_{l_1}, u_{l_2}, \ldots, u_{l_n})$ rešitev problema max-Prilagodljivost 1-Atributov s ceno 0. Za vse $(i, j) \in N$ velja torej $(u_{l_i}, u_{l_j}) \in F$, kar pravzaprav pomeni, da je T cikel v H. Za vse $(i, j) \notin N$ pa velja $l_i \neq l_j$, kar pomeni, da se v T vsako vozlišče iz U pojavi natanko enkrat. Sledi, da je T Hamiltonov cikel v H. □

In še posledica.

Posledica 4.16 ▶ Najmanjša max-Prilagodljivost 1-Atributov *je $\mathcal{NP}$-težek optimizacijski problem.*

4.2.4 Aproksimabilnost

Vsi opisani problemi prilagodljivih atributov so $\mathcal{NP}$-težki. Eden izmed možnih načinov za reševanje $\mathcal{NP}$-težkih problemov je iskanje aproksimacijskih algoritmov. V tem podrazdelku bomo govorili o aproksimabilnosti problemov prilagodljivih atributov. Povedali bomo, kakšne aproksimacijske algoritme je možno sestaviti in kakšnih ni možno. Najprej se bomo ukvarjali z absolutno aproksimabilnostjo problemov prilagodljivih atributov, t.j. z vprašanjem obstoja absolutnih aproksimacijskih algoritmov za probleme, nato pa še z relativno aproksimabilnostjo, t.j. z vprašanjem o obstoju (relativnih) aproksimacijskih algoritmov.

Absolutna aproksimabilnost

Naj bo $x = (G, I, C, p)$ poljubna naloga prilagodljivih atributov. Iz x ustvarimo novo nalogo $x' = (G, I, C', p)$, za katero velja, da vse cene $c_{ij}(a_{ik}, a_{jl})$ pomnožimo s konstanto $k \in \mathbb{Z}^+$. Torej

$$c'_{ij}(a_{ik}, a_{jl}) = k\, c_{ij}(a_{ik}, a_{jl}).$$

Naj bo $S = (S_1, S_2, \ldots, S_n)$ poljubna dopustna rešitev naloge x. Prav gotovo je S tudi dopustna rešitev naloge x'. Naj bosta $f_{sum}(x, S) = \sum_{(i,j)\in E} f_{ij}(S_i, S_j)$ oz. $f_{max}(x, S) = \max_{(i,j)\in E} f_{ij}(S_i, S_j)$, t.j. ceni (glede na sum oz. max prilagodljivost) rešitve S za nalogo x. Velja naslednja trditev.

Lema 4.17 ▶ *Velja $f_{sum}(x', S) = k f_{sum}(x, S)$ in $f_{max}(x', S) = k f_{max}(x, S)$.*

Dokaz. Najprej si oglejmo partikularno prilagodljivost $f'_{ij}(S_i, S_j)$ v primerjavi z $f_{ij}(S_i, S_j)$ za $(i, j) \in E$. Za izračun $f'_{ij}(S_i, S_j)$ uporabimo rešitev naloge $H_{ij} = (S_i \cup S_j, S_i \times S_j)$ problema NAJMANJŠA DODELITEV. (Glej razdelek 4.1.2.) Prav gotovo velja $f'_{ij}(S_i, S_j) = kf_{ij}(S_i, S_j)$. Izračunajmo $f_{sum}(x', S)$. Velja

$$f_{sum}(x', S) = \sum_{(i,j)\in E} f'_{ij}(S_i, S_j) = k \sum_{(i,j)\in E} f_{ij}(S_i, S_j) = kf_{sum}(x, S).$$

Podobno velja tudi

$$f_{max}(x', S) = \max_{(i,j)\in E} f'_{ij}(S_i, S_j) = k \max_{(i,j)\in E} f_{ij}(S_i, S_j) = kf_{max}(x, S).$$

□

S pomočjo pravkar dokazane leme pridemo do naslednjega rezultata.

Trditev 4.18 ▶ *Za nobenega od problemov*

- NAJMANJŠA $\sum$-PRILAGODLJIVOST p-ATRIBUTOV*;*
- NAJMANJŠA max-PRILAGODLJIVOST p-ATRIBUTOV*;*
- NAJMANJŠA $\sum$-PRILAGODLJIVOST 1-ATRIBUTOV*; in*
- NAJMANJŠA max-PRILAGODLJIVOST 1-ATRIBUTOV

ne obstaja absolutni polinomski aproksimacijski algoritem, razen če $\mathcal{P}$=$\mathcal{NP}$.

Dokaz. S protislovjem. Naj bo $x = (G, I, C, p)$ naloga in A polinomski aproksimacijski algoritem z absolutno napako d za kateregakoli od naštetih problemov. Naj bo $A(x)$ rešitev naloge x, ki jo vrne algoritem A, in $f(x, A(x))$ celotna cena prilagajanja rešitve $A(x)$. Za A po predpostavki velja

$$D(x, A(x)) = f(x, A(x)) - f^*(x) \leq d.$$

Ustvarimo novo nalogo $x' = (G, I, C', p)$ iz x, tako da vse cene prilagoditev atributov pomnožimo z $d + 1$. Zaradi leme 4.17 velja $f(x', A(x')) = (d + 1)f(x, A(x))$. Potemtakem je

$$\begin{aligned} D(x', A(x')) &= f(x', A(x')) - f^*(x') = \\ &= (d+1)(f(x, A(x)) - f^*(x)). \end{aligned}$$

To pa mora biti manjše ali enako d, ker je A absolutni aproksimacijski algoritem, kar je možno samo v primeru, da je A natančni algoritem. Takšen algoritem pa v primeru $P \neq \mathcal{NP}$ ne obstaja. □

Pokazali smo, da nobeden od štirih opisanih problemov prilagodljivih atributov ni absolutno aproksimabilen, kar morda niti ne preseneti [ACG+99, Rob02], zato si oglejmo še, kako je z relativno aproksimabilnostjo.

Relativna aproksimabilnost

Preden se lotimo razvoja algoritmov za nek $\mathcal{NP}$-težek optimizacijski problem, je dobro odgovoriti še na vprašanje, ali za dani problem sploh obstaja kak ρ-aproksimacijski algoritem, da velja $\rho < \infty$. Odgovor nanj za probleme prilagodljivih atributov daje naslednja trditev.

Trditev 4.19 ▶ *Za nobenega od problemov*

- Najmanjša $\sum$-Prilagodljivost p-Atributov*;*
- Najmanjša max-Prilagodljivost p-Atributov*;*
- Najmanjša $\sum$-Prilagodljivost 1-Atributov*; in*
- Najmanjša max-Prilagodljivost 1-Atributov

ne obstaja polinomski aproksimacijski algoritem s konstantnim aproksimacijskim faktorjem, razen če $\mathcal{P}$=$\mathcal{NP}$.

Dokaz. Neaproksimabilnost bomo dokazali za vsakega izmed naštetih problemov posebej. To naredimo tako, da predpostavimo, da za problem obstaja ρ-aproksimacijski algoritem. Nato s pomočjo takšnega algoritma ustvarimo algoritem, ki omogoča optimalno reševanje nekega $\mathcal{NP}$-polnega problema v polinomskem času. S tem, v primeru $P \neq \mathcal{NP}$, pridemo do protislovja.

Najprej dokažimo trditev za problem Najmanjša max-Prilagodljivost 1-Atributov. To bomo storili s pomočjo konstrukcije iz dokaza trditve 4.15 ($\mathcal{NP}$-polnost problema). V omenjenem dokazu zamenjamo funkcijo cene prilagoditve atributov z naslednjo

$$c_{ij}(u_k, u_l) = \begin{cases} 1 & (i,j) \in N, (u_k, u_l) \in F \\ 1+\rho & (i,j) \in N, (u_k, u_l) \notin F \\ \infty & (i,j) \notin N, k = l \\ 0 & (i,j) \notin N, k \neq l. \end{cases}$$

Nato iz naloge x za Hamiltonov Cikel ustvarimo nalogo x' za Najmanjša max-Prilagodljivost 1-Atributov. Rešitev naloge x' ima vrednost 1, če in samo če v x obstaja Hamiltonov cikel. V tem primeru ima naslednja najmanjša približna rešitev vrednost $1 + \rho$. V nasprotnem primeru, torej če v x ne obstaja Hamiltonov cikel, ima optimalna rešitev naloge x' vrednost vsaj $1 + \rho$.

Potemtakem s pomočjo polinomskega ρ-aproksimacijskega algoritma A za problem Najmanjša max-Prilagodljivost 1-Atributov lahko sestavimo natančni polinomski algoritem za problem Hamiltonov Cikel na naslednji način: algoritem A poženemo na nalogi x', ki ustreza nalogi x za Hamiltonov Cikel, in odgovorimo z DA, če in samo če A vrne rešitev vrednosti 1.

Posledica neaproksimabilnosti problema NAJMANJŠA max-PRILAGODLJIVOST 1-ATRIBUTOV je neaproksimabilnost splošnejšega problema NAJMANJŠA max-PRILAGODLJIVOST p-ATRIBUTOV. Lahko pa neaproksimabilnost slednjega pokažemo tudi neposredno, tako da v dokazu trditve 4.10 ($\mathcal{NP}$-polnost problema) zamenjamo funkcijo cene prilagoditve atributov z naslednjo

$$c_{0j}(U_i, u_j) = \begin{cases} 1 & u_j \in U_i \\ 1 + \rho p & \text{sicer.} \end{cases}$$

V tem primeru bi nam polinomski ρ-aproksimacijski algoritem omogočal natančno reševanje problema POKRITJE MNOŽICE.

Sedaj dokažimo trditev za problem NAJMANJŠA $\sum$-PRILAGODLJIVOST 1-ATRIBUTOV. Podobno kot zgoraj bomo to storili s pomočjo konstrukcije iz dokaza trditve 4.15 ($\mathcal{NP}$-polnost problema NAJMANJŠA max-PRILAGODLJIVOST 1-ATRIBUTOV). V omenjenem dokazu zamenjamo funkcijo cene prilagoditve atributov z naslednjo

$$c_{ij}(u_k, u_l) = \begin{cases} 1 & (i,j) \in N, (u_k, u_l) \in F \\ 1 + \rho|U| & (i,j) \in N, (u_k, u_l) \notin F \\ \infty & (i,j) \notin N, k = l \\ 0 & (i,j) \notin N, k \neq l. \end{cases}$$

Iz naloge x (omrežje $H = (U, F)$) za HAMILTONOV CIKEL ustvarimo nalogo x' za problem NAJMANJŠA $\sum$-PRILAGODLJIVOST 1-ATRIBUTOV. Rešitev naloge x' ima vrednost $|U|$, če in samo če v x obstaja Hamiltonov cikel. V tem primeru ima naslednja najmanjša približna rešitev vrednost vsaj $|U|(1 + \rho)$. Njen performančni kvocient je torej večji od ρ. Potemtakem bi z ρ-aproksimacijskim algoritmom A lahko ugotovili, ali v x obstaja Hamiltonov cikel, tako da bi preverili, če je vrednost rešitve, ki jo vrne A na nalogi x', enaka $|U|$.

Iz povedanega sledi tudi neaproksimabilnost problema NAJMANJŠA $\sum$-PRILAGODLJIVOST p-ATRIBUTOV, ki pa jo lahko dokažemo tudi neposredno, če v dokazu trditve 4.8 zamenjamo funkcijo cene prilagoditve atributov z naslednjo

$$c_{0j}(U_i, u_j) = \begin{cases} 1 & u_j \in U_i \\ 1 + \rho p n & \text{sicer.} \end{cases}$$

□

Pravkar dokazana trditev je negativen rezultat glede učinkovitega približnega reševanja problemov prilagodljivih atributov. Zato se poraja vprašanje o obstoju enostavnejših vendar aproksimabilnih različic katerega od problemov.

Trikotniška neenakost za cene prilagajanja atributov velja, kadar za vse $i, j, k \in V$, kjer $(i, j) \in E, (i, k) \in E, (k, j) \in E$, velja

$$c_{ij}(a, b) \leq c_{ik}(a, c) + c_{kj}(c, b) \qquad \forall a \in A_i, b \in A_j, c \in A_k.$$

Oglejmo si naslednjo trditev.

Trditev 4.20 ▶ *Za problem* Najmanjša max-Prilagodljivost 1-Atributov*, v katerem velja trikotniška neenakost za cene prilagajanj atributov, ne obstaja ρ-aproksimacijski algoritem z aproksimacijski faktorjem $\rho < 2$, če $\mathcal{P} \neq \mathcal{NP}$.*

Dokaz. V dokazu trditve 4.15 zamenjajmo funkcijo cene prilagoditev atributov z naslednjo

$$c_{ij}(u_k, u_l) = \begin{cases} 1 & (i,j) \in N, (u_k, u_l) \in F \\ 2 & (i,j) \in N, (u_k, u_l) \notin F \\ 2 & (i,j) \notin N, k = l \\ 1 & (i,j) \notin N, k \neq l. \end{cases}$$

Nato iz naloge x za problem Hamiltonov Cikel ustvarimo nalogo x' za problem Najmanjša max-Prilagodljivost 1-Atributov. Za cene prilagajanj atributov v nalogi x' prav gotovo velja trikotniška neenakost. Prav tako velja, da je cena optimalne rešitve naloge x' enaka 1 natanko takrat, ko v x obstaja Hamiltonov cikel. V tem primeru $(2-\epsilon)$-aproksimacijski algoritem vrača rešitev s ceno 1. V nasprotnem primeru pa je cena rešitve 2 in Hamiltonov cikel v x ne obstaja. □

Trditev pravzaprav pove, da je 2-aproksimacijski algoritem najboljše, kar lahko pričakujemo za problem Najmanjša max-Prilagodljivost 1-Atributov, v katerem za cene prilagoditev atributov velja trikotniška neenakost.

4.2.5 Ostale lastnosti

Oglejmo si še nekaj zanimivejših lastnosti problemov prilagodljivih atributov. Naj bo $x = (G, I, C, p)$ poljubna naloga problemov prilagodljivih atributov. Naj bo S poljubna dopustna rešitev naloge x. Zapišimo naslednjo trditev.

Trditev 4.21 ▶ *Velja* $f_{max}(S) \leq f_{sum}(S) \leq |E| f_{max}(S)$.

Trditev lahko dokažemo z uporabo osnovnih lastnosti funkcij $\sum$ in max, zato dokaz opustimo. Zapišimo podobno trditev še za vrednosti optimalnih rešitev. Naj bo S^*_{sum} optimalna rešitev naloge x problema Najmanjša $\sum$-Prilagodljivost p-Atributov in S^*_{max} optimalna rešitev naloge x problema Najmanjša max-Prilagodljivost p-Atributov.

Posledica 4.22 ▶ *Za* S^*_{sum} *in* S^*_{max} *velja*

$$f_{max}(S^*_{max}) \leq f_{max}(S^*_{sum}) \leq f_{sum}(S^*_{sum}) \leq f_{sum}(S^*_{max}).$$

S pomočjo trditve 4.21 in posledice 4.22 dokažimo še naslednje posledice.

Posledica 4.23 ▶ *Naj bo A optimalni algoritem za problem* Najmanjša $\sum$-Prilagodljivost p-Atributov *(oz.* Najmanjša max-Prilagodljivost p-Atributov*). Potemtakem je A tudi $|E|$-aproksimacijski algoritem za problem* Najmanjša max-Prilagodljivost p-Atributov *(oz.* Najmanjša $\sum$-Prilagodljivost p-Atributov*).*

Dokaz. Naj bo x naloga prilagodljivih atributov. Če $S^*_{sum} = A(x)$, očitno velja $f_{max}(S^*_{max}) \leq f_{max}(S^*_{sum}) \leq |E| f_{max}(S^*_{max})$. Prav tako v primeru $S^*_{max} = A(x)$ velja $f_{sum}(S^*_{sum}) \leq f_{sum}(S^*_{max}) \leq |E| f_{sum}(S^*_{sum})$. □

Na prvi pogled se zdi, da zaradi $\mathcal{NP}$-težkosti obeh problemov, takšen algoritem A ne obstaja (razen če $\mathcal{P}$=$\mathcal{NP}$). Vendar nam posledica lahko kljub temu pride prav pri obravnavi poenostavljenih različic splošnih problemov ali pa pri obravnavi superpolinomskih algoritmov.

Zapišimo še naslednjo trditev.

Posledica 4.24 ▶ *Naj bo A ρ-aproksimacijski algoritem za problem* Najmanjša $\sum$-Prilagodljivost p-Atributov *(oz.* Najmanjša max-Prilagodljivost p-Atributov*). Potemtakem je A tudi $\rho|E|$-aproksimacijski algoritem za problem* Najmanjša max-Prilagodljivost p-Atributov *(oz.* Najmanjša $\sum$-Prilagodljivost p-Atributov*).*

Dokaz. Prvi del. Velja $f_{sum}(S^*_{sum}) \leq f_{sum}(A(x)) \leq \rho f_{sum}(S^*_{sum})$. Potemtakem velja $f_{max}(S^*_{max}) \leq f_{max}(A(x)) \leq f_{sum}(A(x)) \leq \rho f_{sum}(S^*_{sum}) \leq \rho f_{sum}(S^*_{max}) \leq \rho|E| f_{max}(S^*_{max})$.

Drugi del. Velja $f_{max}(S^*_{max}) \leq f_{max}(A(x)) \leq \rho f_{max}(S^*_{max})$. Potemtakem velja $f_{sum}(S^*_{sum}) \leq f_{sum}(A(x)) \leq |E| f_{max}(A(x)) \leq \rho|E| f_{max}(S^*_{max}) \leq \rho|E| f_{sum}(S^*_{sum})$. □

4.3 Matematični programi

V tem razdelku se bomo z matematičnim programiranjem lotili reševanja problemov prilagodljivih $p = 1$ atributov, t.j. problemov Najmanjša $\sum$-Prilagodljivost 1-Atributov in Najmanjša max-Prilagodljivost 1-Atributov. Uporabili bomo celoštevilsko kvadratno programiranje (IQP[3]) in celoštevilsko linearno programiranje (ILP[4]). Za reševanje linearnih programov obstaja veliko programskih

[3] *angl. integer quadratic programming*
[4] *angl. integer linear programming*

paketov. Če nek problem definiramo z linearnim programom, ga lahko s takim programskim paketom preprosto praktično rešujemo. Na ta način hitro dobimo prvi vtis o praktični zahtevnosti reševanja problema.

Najprej bomo predstavili skupno notacijo, ki jo bomo uporabljali v vseh programih. Nato bomo za oba problema zapisali celoštevilska kvadratna programa, ki ju dobimo neposredno iz definicij problemov. Nadalje bomo za oba problema zapisali še več celoštevilskih linearnih programov.

4.3.1 Notacija v programih

Notacija, uporabljena v definicijah problemov prilagodljivih atributov, ni neposredno primerna za uporabo v matematičnih programih, zato bomo najprej predstavili primernejšo notacijo. V tem podrazdelku opišemo samo notacijo, ki je skupna vsem matematičnim programom, ostalo bomo po potrebi uvajali sproti.

Naj bo (G, I, C) naloga prilagodljivih atributov. Graf $G = (V, E)$ je v matematičnih programih predstavljen z vrednostmi e_{ij}, ki so definirane kot

$$e_{ij} = \begin{cases} 1 & (i,j) \in E \\ 0 & \text{sicer.} \end{cases}$$

Definirajmo še

$$c_{ikjl} = c_{ij}(a_{ik}, a_{jl}),$$

ki predstavlja ceno prilagoditve atributa $a_{ik} \in A_i$ v $a_{jl} \in A_j$. Indeksa i in j označujeta vozlišča oz. scenarije, indeksa k in l pa atribute v scenarijih i in j zaporedoma.

Odločitvene spremenljivke v programih so x_{ik}, kjer

$$x_{ik} = \begin{cases} 1 & \text{če izberemo atribut } a_{ik} \in A_i \\ 0 & \text{sicer.} \end{cases}$$

Število spremenljivk x_{ik} je enako celotnemu številu atributov, t.j. $\sum_{i \in V} |A_i| = O(nr)$. V vseh programih bodo za vsak $1 \leq i \leq n$ in $1 \leq k \leq |A_i|$ nastopale tudi celoštevilske omejitve

$$x_{ik} \in \{0, 1\}.$$

4.3.2 Kvadratna programa

Najprej se bomo lotili celoštevilskih kvadratnih programov za probleme prilagodljivih atributov, ker so le-ti morda lažji za razumevanje kot linearni. Težava kvadratnih programov je v tem, da se v praksi pogosto upirajo učinkovitemu reševanju.

V tem podrazdelku bomo predstavili celoštevilska kvadratna programa za sum in max prilagodljivost atributov, kjer $p = 1$. V obeh programih bo za $i, j \in V$ nastopal

člen

$$\sum_{1\le k\le|A_i|}\ \sum_{1\le l\le|A_j|} e_{ij}c_{ikjl}x_{ik}x_{jl}. \tag{4.2}$$

V primeru, če $(i,j)\notin E$, je ta vsota enaka 0, ker $e_{ij}=0$. V nasprotnem primeru, če $(i,j)\in E$, in kadar sta hkrati tudi oba x_{ik} in x_{jl} različna od nič – kadar v scenariju s_i izberemo atribut $a_{ik}\in A_i$ in v s_j izberemo $a_{jl}\in A_j$ –, je vsota enaka c_{ikjl}, torej ceni prilagoditve atributa a_{ik} v a_{jl}. Ker gre za izbiro $p=1$ atributov v vsakem od scenarijev, se s tem pravzaprav izračuna partikularna prilagodljivost f_{ij}.

Oglejmo si najprej celoštevilski kvadratni program za sum prilagodljivost.

Algoritem 4.2: IQP za Najmanjša $\sum$-Prilagodljivost 1-Atributov

$$\begin{aligned}
&\text{minimizacija} && \sum_{1\le i,j\le n}\ \sum_{1\le k\le|A_i|}\ \sum_{1\le l\le|A_j|} e_{ij}c_{ikjl}x_{ik}x_{jl} && (4.3a)\\
&\text{pri omejitvah} \quad \forall i && \sum_{1\le k\le|A_i|} x_{ik}=1 && (4.3b)\\
&\qquad\qquad \forall i,k && x_{ik}\in\{0,1\} && (4.3c)
\end{aligned}$$

Osrednji del kriterijske funkcije (4.3a) je sestavljen iz vsote partikularnih prilagodljivosti. (Vsebuje formulo (4.2).) Torej gre za minimizacijo celotne prilagodljivosti. V programu je še n omejitev (4.3b), s katerimi zagotovimo, da se za vsak scenarij s_i izbere natanko en atribut.

Oglejmo si še celoštevilski kvadratni program za max prilagodljivost.

Algoritem 4.3: IQP za Najmanjša max-Prilagodljivost 1-Atributov

$$\begin{aligned}
&\text{minimizacija} && W && (4.4a)\\
&\text{pri omejitvah} \quad \forall i && \sum_{1\le k\le|A_i|} x_{ik}=1 && (4.4b)\\
&\qquad\qquad \forall i,j && W\ge \sum_{1\le k\le|A_i|}\ \sum_{1\le l\le|A_j|} e_{ij}c_{ikjl}x_{ik}x_{jl} && (4.4c)\\
&\qquad\qquad \forall i,k && x_{ik}\in\{0,1\} && (4.4d)
\end{aligned}$$

Celotna prilagodljivost je enaka maksimumu partikularnih prilagodljivosti. Ker maksimuma ne moremo neposredno uporabiti v kriterijski funkciji matematičnih programov, uporabimo preprost trik. Uvedemo dodatno spremenljivko W, ki predstavlja maksimum in hkrati tudi omejitev vseh cen partikularnih prilagajanj c_{ikjl}.

Kriterijska funkcija (4.4a) je tako sestavljena samo iz spremenljivke W, ki pa nastopa še kot zgornja meja v omejitvah (4.4c). S temi omejitvami spremenljivka W za vsak $1 \leq i, j \leq n$ od zgoraj omejuje ceno partikularne prilagodljivosti. Ostale omejitve so enake kot v programu 4.2. Skupno število omejitev (če ne štejemo tudi celoštevilskih) v programu je $n^2 + n$.

4.3.3 Linearni programi

Kot smo že omenili, so kvadratni programi težki za reševanje. V kvadratnih programih se lahko znebimo člena $x_{ik}x_{jl}$, tako da za vsak par x_{ik} in x_{jl} uvedemo novo odločitveno spremenljivko

$$y_{ikjl} = x_{ik}x_{jl},$$

ki pravzaprav pomeni

$$y_{ikjl} = \begin{cases} 1 & \text{če izberemo prilagoditev } a_{ik}\text{v } a_{jl} \\ 0 & \text{sicer.} \end{cases}$$

Seveda tudi za y_{ikjl} veljajo celoštevilske omejitve

$$y_{ikjl} \in \{0, 1\}.$$

Negativni stranski učinek uvedbe y_{ikjl} je pridobitev dodatnih $O(n^2r^2)$ odločitvenih spremenljivk. Poleg tega so potrebne še dodatne omejitve.

Podobno, kot je v kvadratnih programih nastopala funkcija (4.2), bo v linearnih programih nastopala naslednja

$$\sum_{1 \leq k \leq |A_i|} \sum_{1 \leq l \leq |A_l|} e_{ij}c_{ikjl}y_{ikjl}. \tag{4.5}$$

Le-ta seveda zopet predstavlja ceno partikularne prilagoditve. V nadaljevanju predstavimo dva načina linearizacije.

Neposredna linearizacija

V prvem primeru se bomo ravnali po zgledu problema kvadratne dodelitve (QAP[5]) iz [DH02]. Uvedli bomo novo spremenljivko y_{ikjl} in dodali še dve vrsti omejitev. Oglejmo si najprej celoštevilski linearni program za sum prilagodljivost atributov.

[5] *angl. quadratic assignment problem*

Algoritem 4.4: ILP za Najmanjša $\sum$-Prilagodljivost 1-Atributov

$$\text{minimizacija} \quad \sum_{1\le i,j\le n} \sum_{1\le k\le |A_i|} \sum_{1\le l\le |A_j|} e_{ij} c_{ikjl} y_{ikjl} \tag{4.6a}$$

$$\text{pri omejitvah} \quad \forall i \quad \sum_{1\le k\le |A_i|} x_{ik} = 1 \tag{4.6b}$$

$$\sum_{1\le i,j\le n} \sum_{1\le k\le |A_i|} \sum_{1\le l\le |A_j|} y_{ikjl} = n^2 \tag{4.6c}$$

$$\forall i,j,k,l \quad x_{ik} + x_{jl} \ge 2y_{ikjl} \tag{4.6d}$$

$$\forall i,k \quad x_{ik} \in \{0,1\} \tag{4.6e}$$

$$\forall i,j,k,l \quad y_{ikjl} \in \{0,1\} \tag{4.6f}$$

Kriterijska funkcija (4.6a) je podobna kriterijski funkciji (4.3a) iz programa 4.2, le da namesto kvadratnega člena $x_{ik}x_{jl}$ sedaj nastopa samo ena spremenljivka y_{ikjl}. Preden se lotimo razlage novih omejitev (4.6c) in (4.6d), napišimo še celoštevilski linearni program za max prilagodljivost.

Algoritem 4.5: ILP za Najmanjša max-Prilagodljivost 1-Atributov

$$\text{minimizacija} \quad W \tag{4.7a}$$

$$\text{pri omejitvah} \quad \forall i \quad \sum_{1\le k\le |A_i|} x_{ik} = 1 \tag{4.7b}$$

$$\sum_{1\le i,j\le n} \sum_{1\le k\le |A_i|} \sum_{1\le l\le |A_j|} y_{ikjl} = n^2 \tag{4.7c}$$

$$\forall i,j,k,l \quad x_{ik} + x_{jl} \ge 2y_{ikjl} \tag{4.7d}$$

$$\forall i,j \quad W \ge \sum_{1\le k\le |A_i|} \sum_{1\le l\le |A_j|} e_{ij} c_{ikjl} y_{ikjl} \tag{4.7e}$$

$$\forall i,k \quad x_{ik} \in \{0,1\} \tag{4.7f}$$

$$\forall i,j,k,l \quad y_{ikjl} \in \{0,1\} \tag{4.7g}$$

V obeh programih nastopata enaki omejitvi (4.6c) in (4.7c). Za poljubna i in j, kjer $1 \le i,j \le n$, velja $y_{ikjl} = 1$ za natanko en par k in l, kjer $1 \le k \le |A_i|$ in $1 \le l \le |A_j|$; za vse ostale k in l velja $y_{ikjl} = 0$. Upoštevajmo to in zapišimo enačbo (4.6c) (oz. (4.7c)) malce drugače

$$\sum_{1\le i,j\le n} \sum_{1\le k\le |A_i|} \sum_{1\le l\le |A_j|} y_{ikjl} = \sum_{1\le i,j\le n} 1 = n^2.$$

Omejitev pravzaprav pove, da v programu hkrati nastopa n^2 prilagoditev atributov. (Če je $e_{ij} = 0$, se prilagoditev v kriterijski funkciji pravzaprav ne upošteva.)

V obeh programih sta enaki tudi omejitvi (4.6d) in (4.7d). Za njuno razlago si oglejmo naslednjo tabelo 4.6. V tabeli navajamo vrednosti $x_{ik}, x_{jl}, x_{ik} + x_{jl}, 2y_{ikjl}$ in y_{ikjl}. Iz tabele hitro vidimo, da $x_{ik} + x_{jl} \geq 2y_{ikjl}$ pravzaprav predstavlja logični

Tabela 4.6: Omejitev $x_{ik} + x_{jl} \geq 2y_{ikjl}$

x_{ik}	x_{jl}	$x_{ik} + x_{jl}$	$2y_{ikjl}$	y_{ikjl}
0	0	0	0	0
0	1	1	0	0
1	0	1	0	0
1	1	2	2	1

in med x_{ik} in x_{jl}, katerega rezultat je y_{ikjl}.

Skupno število omejitev (brez celoštevilskih) v takšnem načinu linearizacije je $O(n^2r^2 + n + 1)$ v programu 4.4 in $O(n^2r^2 + n^2 + n + 1)$ v programu 4.5.

Učinkovitejša linearizacija

Oglejmo si še alternativni način linearizacije, v katerem nastopa manjše število omejitev in je zato bolj učinkovit. Najprej napišimo celoštevilski linearni program za sum prilagodljivost.

Algoritem 4.6: ILP za Najmanjša $\sum$-Prilagodljivost 1-Atributov

$$\text{minimizacija} \qquad \sum_{1\leq i,j\leq n} \sum_{1\leq k\leq |A_i|} \sum_{1\leq l\leq |A_j|} e_{ij} c_{ikjl} y_{ikjl} \tag{4.8a}$$

$$\text{pri omejitvah} \quad \forall i \qquad \sum_{1\leq k\leq |A_i|} x_{ik} = 1 \tag{4.8b}$$

$$\forall i,k \qquad \sum_{1\leq j\leq n} \sum_{1\leq l\leq |A_j|} y_{ikjl} = n x_{ik} \tag{4.8c}$$

$$\forall j,l \qquad \sum_{1\leq i\leq n} \sum_{1\leq k\leq |A_i|} y_{ikjl} = n x_{jl} \tag{4.8d}$$

$$\forall i,k \qquad x_{ik} \in \{0,1\} \tag{4.8e}$$

$$\forall i,j,k,l \qquad y_{ikjl} \in \{0,1\} \tag{4.8f}$$

Število omejitev (brez celoštevilskih) v programu 4.6 je $O(2nr + n)$. Zapišimo še celoštevilski linearni program za max prilagodljivost.

Algoritem 4.7: ILP za Najmanjša max-Prilagodljivost 1-Atributov

$$\begin{aligned}
&\text{minimizacija} && W && (4.9a)\\
&\text{pri omejitvah} && \forall i && \sum_{1\le k\le |A_i|} x_{ik} = 1 && (4.9b)\\
&&& \forall i,k && \sum_{1\le j\le n}\sum_{1\le l\le |A_j|} y_{ikjl} = nx_{ik} && (4.9c)\\
&&& \forall j,l && \sum_{1\le i\le n}\sum_{1\le k\le |A_i|} y_{ikjl} = nx_{jl} && (4.9d)\\
&&& \forall i,j && W \ge \sum_{1\le k\le |A_i|}\sum_{1\le l\le |A_j|} e_{ij}c_{ikjl}y_{ikjl} && (4.9e)\\
&&& \forall i,k && x_{ik}\in\{0,1\} && (4.9f)\\
&&& \forall i,j,k,l && y_{ikjl}\in\{0,1\} && (4.9g)
\end{aligned}$$

Število omejitev (brez celoštevilskih) v programu 4.7 je $O(n^2+2nr+n)$. V obeh linearnih programih so nove omejitve (4.8c) in (4.8d) oz. (4.9c) in (4.9d). Za vsak i,k vsota

$$\sum_{1\le j\le n}\sum_{1\le l\le |A_j|} y_{ikjl} \tag{4.10}$$

predstavlja število izbranih prilagoditev (v polnem grafu) iz atributa a_{ik} v poljuben atribut. V primeru, da a_{ik} ni v rešitvi (če $x_{ik}=0$), potem ni izbrana nobena od prilagoditev in je zgornja vsota enaka 0. V nasprotnem primeru velja $x_{ik}=1$, izbrane morajo biti vse možne prilagoditve iz a_{ik} v nek izbran atribut a_{jl}, kjer $1\le j\le n$. Teh pa je ravno n. Na ta način utemeljimo omejitvi (4.8c) in (4.9c). Na analogen način utemeljimo še omejitvi (4.8d) in (4.9d) za vsak j in l.

4.3.4 Učinkovitost programov

Velikost programov

Na tem mestu si oglejmo medsebojno primerjavo matematičnih programov. Najprej primerjajmo število odločitvenih spremenljivk. V obeh kvadratnih programih je $O(nr)$ odločitvenih spremenljivk, v linearnih programih pa $O(n^2r^2+nr)$. Ker je vsaka odločitvena spremenljivka celoštevilsko omejena, je ravno toliko tudi celoštevilskih omejitev.

Oglejmo si še, kako je s številom (necelošteviskih) omejitev. Število omejitev za posamezne programe navajamo v tabeli 4.7. V stolpcih sum oz. max je prikazano število omejitev v programih za sum oz. max prilagodljivost, v vrstici z oznako

IQP za celoštevilska kvadratna programa, v vrstici z oznako ILP-A za prvo različico linearizacije in z oznako ILP-B za drugo različico.

Tabela 4.7: Število omejitev v programih

	sum	max
IQP	n	n^2+n
ILP-A	$O(n^2r^2+n+1)$	$O(n^2r^2+n^2+n+1)$
ILP-B	$O(2nr+n)$	$O(2nr+n^2+n)$

Opazimo, da je število omejitev v kvadratnih programih precej manjše kot v linearnih, torej z linearizacijo velikost programa precej naraste. Poleg tega je število omejitev v programih manjše za sum prilagodljivost kot za max prilagodljivost. Opazimo tudi, da je glede števila omejitev različica ILP-B precej boljša kot ILP-A, saj je število omejitev v prvi precej manjše.

Hitrost programov

Zgornji pregled števila omejitev v matematičnih programih že nakazuje, kako se posamezni programi obnašajo v praksi. Vseeno pa smo želeli opraviti praktični preizkus linearnih programov, zato smo zapisali celoštevilčne linearne programe za sum prilagodljivost. V ta namen smo uporabili orodje *GNU Linear Programming Kit* (GLPK) različice 4.7. GLPK omogoča reševanje (celoštevilskih) linearnih programov na uporabniku prijazen način. Linearne programe je potrebno zapisati v programskem jeziku *AMPL*[6], nad katerim nato poženemo *GLPK Solver*. Ta program reši, izpiše vrednost optimalne rešitve, čas reševanja in količino porabljenega pomnilnika. Vse programe smo testirali v okolju Linux Suse 9.2 na računalniku Intel Pentium 1,6 GHz s 512 MB pomnilnika.

Predvsem nas je zanimala primerjava obeh ILP-A in ILP-B linearnih različic programov za problem Najmanjša $\sum$-Prilagodljivost 1-Atributov. Rezultate eksperimentalnega testiranja teh dveh smo zapisali v tabeli 4.8. V stolpcu "velikost" se nahaja velikost matrike (št. vrstic $\times$ št. stolpcev), čas reševanja je podan v sekundah, količina porabljenega pomnilnika (pom) pa v megabajtih.

Naloga, ki smo jo uporabili za testiranje programov, je iz primera, prikazanega na sliki 4.1, z razliko da je v testni nalogi vsak scenarij opisan z enakim številom atributov. To število označimo z $|A_i|$ in ga navajamo v prvem stolpcu tabele 4.8. Programi v jeziku AMPL so navedeni v dodatku.

Kljub temu da gre za zelo majhno nalogo, v kateri nastopa le $n = 5$ scenarijev, sta oba programa že pri $|A_i| = 10$ zelo počasna. Vseeno je iz tabele razvidno, da je ILP-B precej hitrejši od ILP-A, kar je predvsem posledica manjšega števila omejitev v ILP-B. Programa ILP-A zaradi počasnosti sploh nismo pognali na primerih, v katerih $|A_i| \geq 5$.

[6] A Modeling Language for Mathematical Programming (http://www.ampl.com/).

Tabela 4.8: Eksperimentalni rezultati

	ILP-A			ILP-B		
$\|A_i\|$	velikost	čas	pom	velikost	čas	pom
1	31x30	0	0,2	16x30	0	0,2
2	107x110	26	2,1	26x110	0	0,2
3	232x240	167	5,7	36x240	0	0,4
4	407x420	317688	557	46x420	2	0,6
5	632x650			56x650	7	0,8
6	907x930			66x930	21	1,3
7	1226x1260			76x1260	55	1,8
8	1226x1260			86x1640	130	2,6
9	2032x2070			96x2070	278	3,3
10	2507x2550			106x2550	581	3,3

4.4 Reševanje na drevesu

4.4.1 Drevesa

V podrazdelkih 4.2.2 in 4.2.3 smo pokazali, da so problemi prilagodljivih atributov $\mathcal{NP}$-težki. To pomeni, da so možnosti za njihovo učinkovito (polinomsko) reševanje omejene. V takem primeru se je smiselno vprašati, katere od poenostavljenih različic problemov so polinomsko rešljive. Prestrogih omejitev se bomo izogibali, ker bi morda vodile do trivialno rešljivih različic.

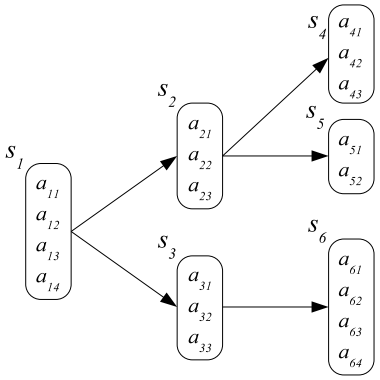
Problemi prilagodljivih atributov so definirani na splošnem grafu prehodov scenarijev. Oglejmo si, kako je z zahtevnostjo reševanja v primeru, kadar je graf prehodov scenarijev drevo. Na koncu podrazdelka 4.2.2 smo pokazali, da sta problem Najmanjša $\sum$-Prilagodljivost p-Atributov kakor Najmanjša max-Prilagodljivost p-Atributov tudi pri tej poenostavitvi $\mathcal{NP}$-težka.

Kaj pa problemi, v katerih za vsak scenarij izberemo natanko en atribut? Izkaže se, da je v primeru drevesnega grafa prehodov scenarijev problema možno natančno reševati v polinomskem času. V nadaljevanju bomo torej opisali polinomska algoritma za optimalno reševanje problemov Najmanjša $\sum$-Prilagodljivost 1-Atributov in Najmanjša max-Prilagodljivost 1-Atributov na nalogah, v katerih je vhodni graf prehodov scenarijev drevo.

Zaradi enostavnejšega opisa algoritmov bomo v nadaljevanju predpostavljali, da scenariji (vozlišča) drevesa niso poljubno označeni, temveč so označeni na naslednji način. Naj bo nek scenarij *korenski scenarij*. Scenarije obiščemo po postopku *iskanja v širino* [CLRS01], pri čemer začnemo s korenskim scenarijem. Vsakič, ko obiščemo nek scenarij, mu priredimo oznako s_i in pri tem i povečamo za ena. Začetna vrednost i naj bo ena. Torej je oznaka korenskega scenarija s_1. Primer tako označenega drevesa se nahaja na sliki 4.9.

Drevo, prikazano na sliki, ima šest vozlišč. Opazimo, da, če potujemo po poti

Slika 4.9: Primer drevesa za problem prilagodljivih atributov



od korena proti poljubnemu listu, oznake vozlišč naraščajo. Za poljubni dve zaporedoma obiskani vozlišči i in j na takšni poti velja $i < j$.

4.4.2 Ogrodje algoritmov

Naj bo (G, I, C) naloga prilagodljivih atributov, v kateri je graf prehodov scenarijev $G = (V, E)$ drevo, in naj bodo vozlišča drevesa G označena na zgoraj opisan način. Lotimo se opisa algoritmov. Označimo s `Sum1FATree` algoritem za reševanje Najmanjša $\sum$-Prilagodljivost 1-Atributov in z `Max1FATree` algoritem za reševanje Najmanjša max-Prilagodljivost 1-Atributov, kjer je vhod obeh algoritmov naloga (G, I, C).

Oba algoritma temeljita na dinamičnem programiranju. Možne rešitve problema se sestavijo iz možnih rešitev podproblemov, te pa nadalje iz možnih rešitev podpodproblemov in tako naprej, dokler ne pridemo do problemov, ki so enostavno rešljivi.

Algoritma potekata iterativno, tako da na vsakem koraku obdelamo eno vozlišče $i \in V$. Listi drevesa G predstavljajo najpreprostejše probleme. Obdelavo začnemo z njimi in nato nadaljujemo z obdelavo vozlišč na poteh v smeri od listov proti korenu drevesa. Naj bo $A = \{(i, j) \in E | i < j\}$. Množica A predstavlja povezave E, usmerjene v smeri od korena proti listom drevesa G. Naj bo $i \in V$ takšno vozlišče, da za vse $(i, j) \in A$ velja, da so vozlišča $j \in V$ že obdelana. V takem primeru je vozlišče i primerno za obdelavo.

Bistvo same obdelave $i \in V$ je, da vsem atributom $a_{ik} \in A_i$ priredimo neko vrednost, ki jo bomo poimenovali utež atributa. Označimo z $w(a_{ik})$ utež atributa a_{ik}. Utež $w(a_{ik})$ predstavlja ceno *neke določene* rešitve za nalogo prilagajanja atributov, v kateri je graf prehodov scenarijev *neko določeno* poddrevo drevesa G. Zaenkrat se zadovoljimo s to ohlapno definicijo, za točno kakšno rešitev in poddrevo gre, pa bomo bolj natančno povedali v nadaljevanju. Na takšen način postopoma računamo uteži atributov, dokler ne pridemo do korenskega scenarija.

Oba algoritma sta skoraj enaka, zato najprej opišimo sestavo enakega dela obeh

algoritmov. Sestavljena sta iz dveh stopenj, t.j. iz inicializacije in dejanskega obdelovanja vozlišč. V inicializaciji se vsem atributom listnih scenarijev priredi utež z vrednostjo 0. V obdelovalni stopnji se na vsakem koraku izbere in obdela neko še neobdelano vozlišče. Razlika med obema algoritmoma je v načinu izračuna uteži atributov. V `Sum1FATree` utež $w(a_{ik})$ izračunamo po naslednji formuli

$$w(a_{ik}) = \sum_{(i,j)\in A} \min_{a_{jl}\in A_j} [c_{ij}(a_{ik}, a_{jl}) + w(a_{jl})]. \tag{4.11}$$

V `Max1FATree` pa po naslednji formuli

$$w(a_{ik}) = \max_{(i,j)\in A} \min_{a_{jl}\in A_j} \max[c_{ij}(a_{ik}, a_{jl}), w(a_{jl})]. \tag{4.12}$$

Formalno sta algoritma `Sum1FATree` in `Max1FATree` zapisana v obliki naslednjega postopka[7].

Algoritem 4.8: `Sum1FATree` in `Max1FATree`

Vhod: Naloga prilagajanja atributov (G, I, C).
Izhod: Vrednost f^*.

```
1.  L := {i|¬∃j : (i,j) ∈ A};
2.  forall i ∈ L do
3.     forall a_ik ∈ A_i do
4.        w(a_ik) := 0;
5.  W := L;
6.  while W ≠ V do
7.     Izberi i ∈ V/W : ∀(i,j) ∈ A : j ∈ W;
8.     forall a_ik ∈ A_i do
9.        Izračunaj w(a_ik);
10.    W := W ∪ {i};
11. end;
12. f* := min_{a_1k ∈ A_1} w(a_1k);
13. return f*;
```

Inicializacijski del najdemo v vrsticah od 1 do 5. Najprej se (vrstica 1) spremenljivki L priredi množica listnih vozlišč oz. scenarijev. Nato se (vrstici 2 in 3) vsakemu atributu listnega vozlišča priredi utež z vrednostjo 0 (vrstica 4). Spremenljivka W označuje obdelana vozlišča; njena začetna vrednost se nastavi na L. Ker vedno skupinsko obdelamo vse atribute posameznega vozlišča, nam W posredno označuje tudi atribute, za katere smo vrednost uteži že izračunali.

Glavna zanka algoritma se nahaja v vrsticah od 6 do 11. Zanka se izvaja, dokler niso vsa vozlišča obdelana (vrstica 6). V telesu zanke se najprej (vrtica 7)

[7]Algoritme bomo pisali v pseudokodi, podobni programskemu jeziku Pascal. Pri tem so pri branju kode pomembni tudi zamiki začetkov programskih vrstic.

izbere takšno neobdelano vozlišče i, katerega vse *usmerjene* povezave vodijo v že obdelana vozlišča. Nato se (vrstici 8 in 9) izračuna utež $w(a_{ik})$ za vse $a_{ik} \in A_i$ izbranega i. Dejanski izračun v vrstici 9 poteka za `Sum1FATree` po formuli (4.11) in za `Max1FATree` po formuli (4.12). Vozlišče i se po obdelavi doda v množico W (vrstica 10). Nato (vrstica 12) izračunamo še vrednost optimalne rešitve problema, ki je enaka najmanjši izmed uteži atributov korena.

Časovna zahtevnost. Ker je G drevo, velja $m = n - 1$. Inicializacija zahteva $O(nr)$ časa. Na vsakem koraku glavne zanke se obdela eno vozlišče. Torej se glavna zanka izvede v največ $O(n)$ korakih. Izbiranje primernega neobdelanega vozlišča i zahteva $O(n)$ časa. Izračunati je potrebno $O(r)$ uteži atributov, od katerih vsak izračun (formula (4.11) oz. (4.12)) zahteva $O(nr)$ časa. Potemtakem je časovna zahtevnost telesa zanke $O(nr^2)$ in celotnega algoritma $O(n^2r^2)$. Tako smo dokazali naslednjo trditev.

Lema 4.25 ▶ *Algoritem* `Sum1FATree` *oz.* `Max1FATree` *je končen in ima polinomsko časovno zahtevnost* $O(n^2r^2)$.

4.4.3 Pravilnost algoritmov

Naj bo $\mathcal{N}[i]$ zaprta soseščina vozlišča $i \in V$ glede na povezave A, torej $\mathcal{N}[i] = \{i\} \cup \{j|(i,j) \in A\}$. Označimo s $T(U)$ induciran podgraf grafa G. Potemtakem je $T(\mathcal{N}[i])$ inducirano drevo s korenom i in vozlišči, ki so dosegljiva iz i s povezavami v A.

Naj bo $(a_{ik}, \dots)$ takšna rešitev (sum ali max prilagodljivosti) na grafu prehodov scenarijev $T(\mathcal{N}[i])$, kjer v vozlišču i fiksiramo izbiro atributa na a_{ik}. Pravzaprav naj bo $(a_{ik}, \dots)$ celo optimalna takšna rešitev, kar pomeni, da v i izberemo a_{ik}, v vseh ostalih vozliščih drevesa $T(\mathcal{N}[i])$ pa atribute izberemo tako, da je celotna cena prilagajanja najmanjša.

Precej ohlapno smo že definirali, da utež $w(a_{ik})$ predstavlja ceno *neke določene* rešitve za nalogo prilagajanja atributov, v kateri je graf prehodov scenarijev *neko določeno* poddrevo drevesa G. Povejmo to sedaj bolj natančno. Utež $w(a_{ik})$ predstavlja ceno rešitve $(a_{ik}, \dots)$ za nalogo sum oz. max prilagodljivosti, tako da velja naslednje

- graf $T(\mathcal{N}[i])$ prehodov scenarijev, kjer $i \in V$, je induciran podgraf drevesa G;
- v rešitvi $(a_{ik}, \dots)$ za scenarij s_i fiksiramo izbiro atributa a_{ik};
- v vseh ostalih scenarijih pa izberemo atribute tako, da je $f_{sum}(a_{ik}, \dots)$ oz. $f_{max}(a_{ik}, \dots)$ najmanjša.

Pokažimo, da povedano velja za `Sum1FATree`. Naj bo $f_{sum}(T(\mathcal{N}[i]), (a_{ik}, \dots))$ cena rešitve $(a_{ik}, \dots)$ v nalogi sum prilagajanja, kjer je graf prehodov scenarijev $T(\mathcal{N}[i])$, in naj bo $w(a_{ik})$ utež, izračunana z algoritmom `Sum1FATree`. Velja naslednja lema.

Lema 4.26 ▶ *Za vse $a_{ik} \in A_i$, kjer $i \in V$, velja $w(a_{ik}) = f_{sum}(\mathcal{N}[i]), (a_{ik}, \dots))$.*

Dokaz. Dokaz izvedemo z indukcijo, glede na velikost poddreves $T(\mathcal{N}[i])$ v G. Začnemo s poddrevesi, sestavljenimi iz enega samega vozlišča, t.j. iz listov drevesa G. Nato potujemo po poteh od listov proti korenu drevesa G in dodajamo obiskana vozlišča.

Začetni korak. Naj bo i poljuben list drevesa G in naj bo $a_{ik} \in A_i$ nek njegov atribut. Oglejmo si vrednost uteži $w(a_{ik})$. V algoritmu `Sum1FATree` se uteži $w(a_{ik})$ atributov, ki pripadajo listnim scenarijem, inicializirajo na 0, torej $w(a_{ik}) = 0$. Po drugi strani pa velja $T(\mathcal{N}[i]) = (\{i\}, \emptyset)$. Potemtakem $f_{sum}(T(\mathcal{N}[i]), (a_{ik})) = 0$. Torej prav gotovo velja $w(a_{ik}) = f_{sum}(T(\mathcal{N}[i]), (a_{ik}))$.

Induktivni korak. Sedaj si oglejmo vrednost uteži $w(a_{ik})$ atributa $a_{ik} \in A_i$, kjer je i vozlišče, ki ni list drevesa G. Po induktivni predpostavki za vse $a_{jl} \in A_j$, kjer $(i, j) \in A$, velja $w(a_{jl}) = f_{sum}(T(\mathcal{N}[j]), (a_{jl}, \dots))$. V `Sum1FATree` poteka izračun $w(a_{ik})$ po formuli (4.11). Vstavimo $w(a_{jl})$ v to formulo in dobimo

$$
\begin{aligned}
w(a_{ik}) &= \sum_{(i,j)\in A} \min_{a_{jl}\in A_j} [c_{ij}(a_{ik}, a_{jl}) + f_{sum}(T(\mathcal{N}[j]), (a_{jl}, \dots))] = && \text{(4.13a)}\\
&= \sum_{(i,j)\in A} \min_{a_{jl}\in A_j} f_{sum}(T(\{i\} \cup \mathcal{N}[j]), (a_{ik}, a_{jl}, \dots)) = && \text{(4.13b)}\\
&= \sum_{(i,j)\in A} f_{sum}(T(\{i\} \cup \mathcal{N}[j]), (a_{ik}, \dots)) = && \text{(4.13c)}\\
&= f_{sum}(T(\mathcal{N}[i]), (a_{ik}, \dots)). && \text{(4.13d)}
\end{aligned}
$$

V enačbi (4.13a) gre za fiksiranje izbire a_{ik} v vozlišču i, poleg tega pa nastopa še rešitev $(a_{jl}, \dots)$ za $T(\mathcal{N}[j])$; oboje skupaj predstavlja rešitev $(a_{ik}, a_{jl}, \dots)$ za drevo $T(\{i\} \cup \mathcal{N}[j])$. V (4.13b) izmed vseh $a_{jl} \in A_j$ izberemo tistega z najmanjšo (in s tem tudi optimalno) ceno prilagajanja. V (4.13c) seštejemo cene prilagajanj rešitev $(a_{ik}, \dots)$ za vsa poddrevesa $T(\{i\} \cup \mathcal{N}[j]$, kjer $(i, j) \in A$. To pa v bistvu predstavlja celotno prilagodljivost rešitve $(a_{ik}, \dots)$ za poddrevo $T(\mathcal{N}[i])$. □

Podobno lemo imamo tudi za `Max1FATree`. Naj bo $f_{max}(T(\mathcal{N}[i]), (a_{ik}, \dots))$ cena rešitve $(a_{ik}, \dots)$ v nalogi max prilagajanja, kjer je graf prehodov scenarijev $T(\mathcal{N}[i])$, in naj bo $w(a_{ik})$ utež, izračunana z algoritmom `Max1FATree`. Velja naslednja lema.

Lema 4.27 ▶ *Za vse $a_{ik} \in A_i$, kjer $i \in V$, velja $w(a_{ik}) = f_{max}(T(\mathcal{N}[i]), (a_{ik}, \dots))$.*

Dokaz. Dokaz je analogen dokazu leme 4.26. Začetni korak indukcije je enak, zato si oglejmo le induktivni korak. Najprej si oglejmo vrednost uteži $w(a_{ik})$ atributa $a_{ik} \in A_i$, kjer je i vozlišče, ki ni list drevesa G. Po induktivni hipotezi za vse $a_{jl} \in A_j$, kjer $(i,j) \in A$, velja $w(a_{jl}) = f_{max}(T(\mathcal{N}[j]), (a_{jl}, \dots))$. V `Max1FATree` poteka izračun $w(a_{ik})$ po formuli (4.12). Vstavimo $w(a_{jl})$ v to formulo in dobimo

$$\begin{aligned} w(a_{ik}) &= \max_{(i,j)\in A} \min_{a_{jl}\in A_j} \max[c_{ij}(a_{ik}, a_{jl}), f_{max}(T(\mathcal{N}[j]), (a_{jl}, \dots))] = && \text{(4.14a)} \\ &= \max_{(i,j)\in A} \min_{a_{jl}\in A_j} f_{max}(T(\{i\} \cup \mathcal{N}[j]), (a_{ik}, a_{jl}, \dots)) = && \text{(4.14b)} \\ &= \max_{(i,j)\in A} f_{max}(T(\{i\} \cup \mathcal{N}[j]), (a_{ik}, \dots)) = && \text{(4.14c)} \\ &= f_{max}(T(\mathcal{N}[i]), (a_{ik}, \dots)). && \text{(4.14d)} \end{aligned}$$

Sklepanje v posameznih korakih je podobno kot v omenjenem dokazu. □

Glede na lemo 4.26 oz. 4.27 imamo naslednjo posledico.

Posledica 4.28 ▶ *Za problem* Najmanjša $\sum$-Prilagodljivost 1-Atributov *(oz.* Najmanjša max-Prilagodljivost 1-Atributov*) algoritem* ***Sum1FATree*** *(oz.* ***Max1FATree****) vrača natančne rešitve nalog* (G, I, C)*, v katerih je* G *drevo.*

Dokaz. Po lemi 4.26 (oz. 4.27) velja, da je za vse $a_{1k} \in A_1$, kjer je $1 \in V$ koren drevesa G, utež $w(a_{1k})$ enaka ceni optimalne rešitve problema Najmanjša $\sum$-Prilagodljivost 1-Atributov (oz. Najmanjša max-Prilagodljivost 1-Atributov) na drevesnem grafu prehodov scenarijev, v primeru, da izbiro atributa v korenu fiksiramo na a_{1k}. V `Sum1FATree` (oz. `Max1FATree`) velja $f^* = \min_{a_{1k}\in A_1} w(a_{1k})$. Torej je s tem f^* tudi optimalna. □

Iz leme 4.25 in posledice 4.28 sledi naslednja trditev.

Trditev 4.29 ▶ *Za naloge* (G, I, C)*, kjer je* G *drevo, problema* Najmanjša $\sum$-Prilagodljivost 1-Atributov *(oz.* Najmanjša max-Prilagodljivost 1-Atributov*) je* ***Sum1FATree*** *(oz.* ***Max1FATree****) natančni polinomski algoritem.*

4.4.4 Rekurzivna oblika algoritmov

Algoritma `Sum1FATree` in `Max1FATree` lahko zapišemo tudi z uporabo rekurzije. Zapis v rekurzivni obliki je morda elegantnejši, poleg tega se tako znebimo še spremenljivke W, v kateri se hranijo neobdelana vozlišča. Vozlišče je obdelano, ko so izračunane uteži vseh njegovih atributov; te izračunamo iz uteži atributov, ki pripadajo potomcem vozlišča. To pomeni, da je potrebno v drevesu ubrati takšen način obdelave vozlišč, ki najprej obdela vse potomce nekega vozlišča, nato šele vozlišče samo. Rekurzivni algoritem temelji na načelu *iskanja v globino* [CLRS01], vozlišča

pa obdelujemo v *postfiksnem* vrstnem redu. Rekurzivni postopek je zapisan v obliki naslednjega algoritma.

Algoritem 4.9: Rekurzivna oblika `Sum1FATree` oz. `Max1FATree`

Vhod: Naloga (G, I, C) prilagodljivih atributov.
Izhod: Vrednost f^*.

```
1.  procedure ObdelajVozlišče(i: vertex);
2.  if (¬∃j : (i, j) ∈ A) then
3.     forall a_ik ∈ A_i do w(a_ik) := 0;
4.  else
5.     forall (i, j) ∈ A do
6.        ObdelajVozlišče(j);
7.     forall a_ik ∈ A_i do
8.        Izračunaj w(a_ik);
9.     return min_{a_ik ∈ A_i} w(a_ik);
10. end;
```

Na kratko opišimo pomembnejše dele algoritma. Pogoj v vrstici 2 je ustavitveni pogoj rekurzije in je resničen, če je i list drevesa G. V tem primeru (vrstica 3) vrednosti uteži atributov $a_{ik} \in A_i$ nastavimo na 0. V nasprotnem primeru se izvede del programa od 5. do 9. vrstice. V tem delu se najprej (vrstici 5 in 6) obdelajo potomci vozlišča i, nato se (vrstici 7 in 8) izračunajo uteži atributov, ki pripadajo vozlišču i. V vrstici 9 se vrne najmanjša utež, t.j. vrednost optimalne rešitve na poddrevesu $T(\mathcal{N}[i])$. Sam postopek poženemo s klicem ObdelajVozlišče(1), kjer je 1 koren grafa G.

4.4.5 Izpis rešitve

Opazimo, da algoritma `Sum1FATree` in `Max1FATree` pravzaprav vrneta le vrednost optimalne rešitve problema, ne vrneta pa same rešitve naloge. Rešujeta torej nekonstrukcijsko obliko problemov. Oba algoritma lahko spremenimo tako, da optimalno rešitev $(a_1, a_2, \ldots, a_n)$ tudi konstruirata.

V algoritmih `Sum1FATree` in `Max1FATree` je potrebno za vsak atribut a_{ik} poleg uteži $w(a_{ik})$ dodatno beležiti še atribut, ki ga je potrebno izbrati v s_j, če sledimo povezavi $(i, j) \in A$. To beležimo s spremenljivko b_{ikj}, katere pomen je: če v s_i izberemo a_{ik}, potem je treba v s_j izbrati b_{ikj}. V algoritmu `Sum1FATree` namesto vrstic 8 in 9 (oz. 7 in 8 v rekurzivni obliki) zapišemo naslednji del programa.

Algoritem 4.10: Konstrukcija rešitve

```
1.  forall a_ik ∈ A_i do
2.     w(a_ik) := 0;
3.     forall (i, j) ∈ A do
4.        (w, b_ikj) := min_{a_jl ∈ A_j} [c_ij(a_ik, a_jl) + w(a_jl)];
5.        w(a_ik) := w(a_ik) + w;
6.     end;
7.  end;
```

V vrstici 4 se izračuna (w, b_{ikj}), kjer w predstavlja rezultat in b_{ikj} argument funkcije min. Velja $w = f_{sum}(T(\{i\} \cup \mathcal{N}[j]), (a_{ik}, b_{ikj}, \dots))$. Ker je b_{ikj} atribut, pri katerem je vrednost izraza znotraj min najmanjša, lahko zapišemo $(a_{ik}, b_{ikj}, \dots) = (a_{ik}, \dots)$. Podobno spremenimo `Max1FATree`, le da v tem primeru v vrsticah 4 in 5 namesto vsote uporabimo maksimum.

Atribut, ki ga je potrebno izbrati v korenskem scenariju, izračunamo z $a_1 = \arg\min_{a_{1k} \in A_1} w(a_{1k})$. Optimalno rešitev $(a_1, a_2, \dots, a_n)$ lahko enostavno skonstruiramo z rekurzivnim sledenjem vrednostim b_{ikj} od korena do listov.

Algoritem 4.10 je prostorsko precej potraten. Za vsak atribut $a_{ik} \in A_i$ si je potrebno zabeležiti $\delta(i)$ vrednosti b_{ikj}, to pa je reda $O(nr)$ za vsak scenarij in $O(n^2 r)$ za ves algoritem. Zato je precej bolj smiselno vrednosti b_{ikj} izračunavati sproti po potrebi. To naredi naslednji rekurzivni algoritem, ki ga poženemo, ko smo z algoritmom 4.8 že izračunali vrednosti uteži vseh atributov.

Algoritem 4.11: Izpis rešitve

```
1.  procedure IzpisRešitve(i: vertex, a_i: attribute)
2.  print "Scenarij: ", i, " atribut: ", a_i;
3.  forall (i, j) ∈ A do
4.     a_j := arg min_{a_jl ∈ A_j} [c_ij(a_i, a_jl) + w(a_jl)];
5.     IzpisRešitve(j, a_j);
6.  end;
```

V vrstici 2 se izpiše atribut, ki je izbran v scenariju i. Nato se (vrstici 3 in 4) izračunajo atributi, ki jih izberemo v scenarijih, ki so potomci i. V vrstici 5 pa se za vse potomce vozlišča i izvede rekurzivni klic. Če želimo izpisati celotno optimalno rešitev, kličemo IzpisRešitve$(1, a_1)$, kjer je $a_1 = \arg\min_{a_{1k} \in A_1}$.

4.5 Reševanje na splošnem grafu

V tem delu bomo razvili algoritem za optimalno reševanje problemov prilagajanja $p = 1$ atributov na splošnem grafu prehodov scenarijev. Ker sta obe vrsti prilagodljivosti $\mathcal{NP}$-težka optimizacijska problema, je malo verjetno pričakovati polinom-

ski algoritem. Vseeno bomo navedli nekaj nasvetov, ki omogočajo učinkovitejše reševanje splošnega problema.

4.5.1 Izločanje scenarija z osamljenim atributom

Najprej si oglejmo poseben primer naloge problema prilagodljivih atributov, v kateri je vsaj eden od scenarijev opisan z natanko enim atributom. Takšen atribut bomo poimenovali tudi *osamljeni* atribut. Izkaže se, da je scenarije z osamljenimi atributi mogoče iz naloge izločiti in na ta način ustvariti enakovredno nalogo. Iz optimalne rešitve enakovredne naloge preprosto dobimo optimalno rešitev originalne naloge tako, da rešitvi originalne naloge dodamo vse izločene osamljene atribute.

Oglejmo si postopek izločanja enega scenarija z osamljenim atributom bolj natančno. Naj bo (G, I, C) naloga problema prilagodljivih atributov, ki vsebuje osamljene atribute in naj bo s_i poljuben scenarij, tako da velja $A_i = \{a_i\}$. Z izločitvijo s_i graf $G = (V, E)$ razpade na kvečjemu $\delta(i)$ povezanih delov. Označimo jih z $G_{ij} = (V_{ij}, E_{ij})$, kjer $j \in \mathcal{N}(i)$. Vsak od G_{ij} je induciran podgraf v G z vozlišči $V_{ij} \subset V$, ki so v $G/\{i\}$ dosegljiva iz j.

Pri tem pravzaprav ni nujno, da so si vsi grafi G_{ij} med seboj različni, zato definirajmo $\mathcal{G}(i) = \{G_{ij} | j \in \mathcal{N}(i)\}$, ki vsebuje le različne G_{ij}. Označimo z I_{ij} množico scenarijev, ki pripadajo vozliščem V_{ij}, in s C_{ij} množico funkcij c_{uv}, kjer $(u, v) \in E_{ij}$. Slednjo bomo natančneje definirali v nadaljevanju. Torej z izločanjem scenarija s_i nalogo (G, I, C) razbijemo na kvečjemu $\delta(i)$ nalog (G_{ij}, I_{ij}, C_{ij}), kjer $G_{ij} \in \mathcal{G}$.

Z izločitvijo i iz G seveda iz G posredno izločimo tudi vse povezave (i, j), kjer $j \in \mathcal{N}(i)$. Zatorej je potrebno cene prilagajanja c_{ij} nekako upoštevati v nastalih nalogah (G_{ij}, I_{ij}, C_{ij}). Glede na velikost G_{ij} si ločeno oglejmo naslednji dve možnosti:

- G_{ij} je sestavljen le iz enega vozlišča;
- G_{ij} je sestavljen iz več kot enega vozlišča.

V primeru prve možnosti gre pravzaprav za enostavno nalogo $((\{j\}, \emptyset), \{s_j\}, \emptyset)$; optimalna rešitev te naloge je izbira poljubnega atributa $a_{jl} \in A_j$. Potemtakem lahko izberemo atribut $a_{jl} \in A_j$, katerega cena prilagajanja na izločeni osamljeni atribut a_i je najmanjša, t.j.

$$a_j = \arg\min_{a_{jl} \in A_j} c_{ij}(a_i, a_{jl}).$$

Prav gotovo je takšna izbira atributa a_j v s_j optimalna izbira za nalogo (G, I, C).

V drugem primeru, kjer G_{ij} vsebuje več kot eno vozlišče, pa cene prilagajanja a_i na atribute $a_{jl} \in A_j$ porazdelimo na cene prilagajanja a_{jl} na $a_{ef} \in A_e$, kjer je $e \in \mathcal{N}_{G_{ij}}(j)$ poljubno j-ju sosednje vozlišče v G_{ij}, ali povedano preprosteje, c_{ij} porazdelimo na c_{je}, kjer $e \in \mathcal{N}_{G_{ij}}(j)$. Pri tem je pomembno, da za porazdelitev

izberemo samo enega (poljubnega) izmed j-ju sosednjih vozlišč. Natančneje, naj bo $e \in \mathcal{N}_{G_{ij}}(j)$. Za $a_{jl} \in A_j$ in $a_{ef} \in A_e$ definirajmo

$$c'_{je}(a_{jl}, a_{ef}) = c_{ij}(a_i, a_{jl}) + c_{je}(a_{jl}, a_{ef}),$$

če gre za problem sum prilagodljivosti atributov in

$$c'_{je}(a_{jl}, a_{ef}) = \max\left[c_{ij}(a_i, a_{jl}), c_{je}(a_{jl}, a_{ef})\right],$$

če gre za problem max prilagodljivosti atributov. Za vse ostale $(u, v) \in E$ pa naj velja $c'_{uv} = c_{uv}$. Definirajmo še $C_{ij} = \{c'_{uv} | (u, v) \in E_{ij}\}$. Na ta način je naloga (G_{ij}, I_{ij}, C_{ij}) popolnoma definirana.

Cena prilagajanja atributa $a_{jl} \in A_j$ v nalogi (G_{ij}, I_{ij}, C_{ij}) torej upošteva tudi $c_{ij}(a_i, a_{jl})$, t.j. ceno prilagajanja izločenega osamljenega atributa a_i na a_{jl}, ki je bila izločena iz naloge (G, I, C). Potemtakem je optimalna rešitev naloge (G_{ij}, I_{ij}, C_{ij}) vsebovana v optimalni rešitvi naloge (G, I, C). Pokažimo to bolj formalno.

Iz rešitev $S_{ij} = (\dots, a_j, \dots)$ nalog (G_{ij}, I_{ij}, C_{ij}), kjer $G_{ij} \in \mathcal{G}(i)$, lahko z združevanjem enostavno sestavimo rešitev $(a_1, a_2, \dots, a_n)$ naloge (G, I, C). Za I_{ij} in I_{ik}, kjer $G_{ij}, G_{ik} \in \mathcal{G}(i)$ in $j \neq k$, velja $I_{ij} \cap I_{ik} = \emptyset$, torej tudi $S_{ij} \cap S_{ik} = \emptyset$. Poleg tega velja še $I = \{s_i\} \cup \bigcup_{G_{ij} \in \mathcal{G}(i)} I_{ij}$ in posledično torej

$$(a_1, a_2, \dots, a_n) = (a_i) \cup \bigcup_{G_{ij} \in \mathcal{G}(i)} S_{ij}.$$

Če $|I_{ij}| = 1$, potem $S_{ij} = (a_j)$. Za rešitev S_{ij} naloge (G_{ij}, I_{ij}, C_{ij}) definirajmo

$$f_{ij}(S_{ij}) = \begin{cases} c_{ij}(a_i, a_j) & |S_{ij}| = 1 \\ f_{sum}(S_{ij}) & |S_{ij}| \geq 2, \end{cases}$$

kjer je $f_{sum}(S_{ij})$ izračunana glede na nalogo (G_{ij}, S_{ij}, C_{ij}).

Cena rešitve $(a_1, a_2, \dots, a_n)$, ki je sestavljena iz rešitev S_{ij}, je enaka vsoti $f_{ij}(S_{ij})$, kjer $G_{ij} \in \mathcal{G}(i)$. Zapišimo to v obliki naslednje leme.

Lema 4.30 ▶ *Naj bo (G, I, C) naloga prilagodljivih atributov, kjer $G = (V, E)$. Naj bo $i \in V$ in S_{ij} rešitev naloge (G_{ij}, I_{ij}, C_{ij}), kjer $G_{ij} \in \mathcal{G}(i)$. Naj bo $(a_1, a_2, \dots, a_n)$ rešitev naloge (G, I, C) sestavljena iz rešitev S_{ij}. Velja*

$$f_{sum}(a_1, a_2, \dots, a_n) = \sum_{G_{ij} \in \mathcal{G}(i)} f_{ij}(S_{ij}).$$

Dokaz.

$$\begin{aligned}
f(a_1, a_2, \ldots, a_n) &= \sum_{(u,v)\in E} c_{uv}(a_u, a_v) = \\
&= \sum_{G_{ij}\in\mathcal{G}(i)} \sum_{(u,v)\in E_{ij}} c_{uv}(a_u, a_v) + \sum_{j\in\mathcal{N}(i)} c_{ij}(a_i, a_j) = \\
&= \sum_{G_{ij}\in\mathcal{G}(i)} \sum_{(u,v)\in E_{ij}} c_{uv}(a_u, a_v) + \sum_{j\in\mathcal{N}(i),\delta(j)\geq 2} c_{ij}(a_i, a_j) + \\
&\quad + \sum_{j\in\mathcal{N}(i),\delta(j)=1} c_{ij}(a_i, a_j) = \\
&= \sum_{G_{ij}\in\mathcal{G}(i)} \sum_{(u,v)\in E_{ij}} c'_{uv}(a_u, a_v) + \sum_{G_{ij}\in\mathcal{G}(i),|I_{ij}|=1} c_{ij}(a_i, a_j) = \\
&= \sum_{G_{ij}\in\mathcal{G}(i),\delta(j)\geq 2} f_{ij}(S_{ij}) + \sum_{G_{ij}\in\mathcal{G}(i),|I_{ij}|=1} f_{ij}(S_{ij}) = \\
&= \sum_{G_{ij}\in\mathcal{G}(i)} f_{ij}(S_{ij}).
\end{aligned}$$

□

Če imamo opravka z max prilagodljivostjo atributov, f_{ij} definirajmo raje kot

$$f_{ij}(S_{ij}) = \begin{cases} c_{ij}(a_i, a_j) & |S_{ij}| = 1 \\ f_{max}(S_{ij}) & |S_{ij}| \geq 2, \end{cases}$$

kjer je $f_{max}(S_{ij})$ izračunana glede na nalogo (G_{ij}, S_{ij}, C_{ij}). Analogna lema velja tudi v tem primeru, le da v dokazu namesto $\sum$ nastopa max.

Lema 4.31 ▶ *Naj bo (G, I, C) naloga prilagodljivih atributov, kjer $G = (V, E)$. Naj bo $i \in V$ in S_{ij} rešitev naloge (G_{ij}, I_{ij}, C_{ij}), kjer $G_{ij} \in \mathcal{G}(i)$. Naj bo $(a_1, a_2, \ldots, a_n)$ rešitev naloge (G, I, C) sestavljena iz rešitev S_{ij}. Velja*

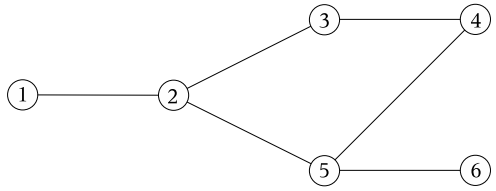
$$f_{max}(a_1, a_2, \ldots, a_n) = \max_{G_{ij}\in\mathcal{G}(i)} f_{ij}(S_{ij}).$$

Naj bo $(a_1, a_2, \ldots, a_n)$ takšna rešitev naloge (G, I, C), da so vrednosti vseh $f_{ij}(S_{ij})$, kjer $G_{ij} \in \mathcal{G}(i)$, najmanjše, potem je tudi vrednost $f_{sum}(a_1, a_2, \ldots, a_n)$ (oz. $f_{max}(a_1, a_2, \ldots, a_n)$) najmanjša. Naj bo S_{ij} optimalna rešitev naloge (G_{ij}, I_{ij}, C_{ij}), kjer $G_{ij} \in \mathcal{G}(i)$ in $|I_{ij}| \geq 2$. V primeru $|I_{ij}| = 1$, t.j. $S_{ij} = \{a_j\}$, pa naj velja $a_j = \arg\min_{a_{jl}\in A_j} c_{ij}(a_i, a_{jl})$; takšna izbira a_j prav gotovo minimizira $f_{ij}(S_{ij})$. Torej velja naslednja trditev.

Trditev 4.32 ▶ *Optimalno rešitev naloge (G, I, C) lahko sestavimo iz optimalnih rešitev nalog (G_{ij}, I_{ij}, C_{ij}), kjer $G_{ij} \in \mathcal{G}(i)$, ki jih dobimo z izločanjem scenarija s_i z osamljenim atributom.*

Primer. Oglejmo si primer izločanja scenarija z osamljenim atributom. Naj bo (G, I, C) naloga prilagodljivih atributov, kjer je $G = (V, E)$ graf prehodov scenarijev, prikazan na sliki 4.10. Naj scenarij s_2 vsebuje osamljen atribut, torej $A_2 = \{a_{21}\}$. Ostali scenariji pa naj vsebujejo poljubno število atributov.

Slika 4.10: Izločanje scenarija z osamljenim atributom.



Vozlišče 2 lahko izločimo iz G. Velja $\mathcal{N}(2) = \{1, 3, 5\}$, torej G razpade na G_{21}, G_{23} in G_{25}, vendar $G_{23} = G_{25}$, zato $\mathcal{G}(2) = \{G_{21}, G_{23}\}$. Porazdeliti je potrebno c_{12}, c_{23} in c_{35}. c_{12} ne moremo porazdeliti, ker G_{21} ne vsebuje nobene povezave, vendar v s_1 lahko izberemo poljuben atribut, zato izberemo $a_1 = \arg\min_{a_{1l} \in A_1} c_{12}(a_{1l}, a_{21})$. c_{23} porazdelimo na c_{34}, torej za $a_{3k} \in A_3$ in $a_{4f} \in A_4$ velja

$$c'_{34}(a_{3k}, a_{4f}) = c_{23}(a_{21}, a_{3k}) + c_{34}(a_{3k}, a_{4f}).$$

Medtem ko c_{25} lahko porazdelimo na c_{45} bodisi c_{56}. Izberimo npr. c_{45}, torej za $a_{4k} \in A_4$ in $a_{5f} \in A_5$ definirajmo

$$c'_{45}(a_{4k}, a_{5f}) = c_{25}(a_{21}, a_{5f}) + c_{45}(a_{4k}, a_{5f}).$$

Iz rešitve $S_{21} = \{a_1\}$ za (G_{21}, I_{21}, C_{21}) in rešitve $S_{23} = S_{25} = \{a_3, a_4, a_5, a_6\}$ za (G_{23}, I_{23}, C_{23}) enostavno sestavimo rešitev $\{a_1, a_2, \ldots, a_6\}$ za (G, I, C).

Če graf G vsebuje več scenarijev z osamljenimi atributi, lahko seveda postopek izločanja ponovimo na vseh takih scenarijih; na ta način lahko enostavno sestavimo rekurzivni algoritem. Lahko pa hkrati izločimo kar vsa vozlišča $U \subseteq V$, ki vsebujejo osamljeni atribut. Naj bo graf $H = (V/U, F)$ induciran na G. Definirajmo $\mathcal{G}(J) = \{G_i | i \in \mathcal{N}(J)/J\}$[8], kjer je $G_i = (V_i, E_i)$ induciran na H in so $V_i \subset V$ vozlišča dosegljiva iz i v grafu H. Tudi v tem primeru lahko iz optimalnih rešitev S_i nalog (G_i, S_i, C_i), kjer $G_i \in \mathcal{G}(J)$, sestavimo optimalno rešitev naloge (G, I, C).

Pri tem morda nastane težava z izločenimi scenariji i, katerih sosednji scenariji $j \in \mathcal{N}(i)$ vsi vsebujejo osamljeni atribut. V tem primeru lahko $c_{ij}(a_i, a_j)$ porazdelimo na poljubno c_{uv}, kjer $(u, v) \in F$, t.j

$$c'_{uv}(a_{uk}, a_{vl}) = c_{uv}(a_{uk}, a_{vl}) + c_{ij}(a_i, a_j).$$

[8] $\mathcal{N}(J) = \cup_{j \in J} \mathcal{N}(j)$, t.j. odprta soseščina vseh vozlišč iz J.

4.5.2 Izločanje poljubnega scenarija

Izločanje scenarijev z osamljenimi atributi omogoča enostavno delitev naloge problema prilagodljivih atributov na več manjših delov. Vprašajmo se, kaj lahko storimo v primeru, da graf prehodov scenarijev ne vsebuje nobenega scenarija z osamljenim atributom ali če smo vse takšne scenarije že izločili.

Fiksiranje atributov

Naj bo (G, I, C) naloga prilagodljivih atributov, iz katere želimo izločiti scenarij $s_i \in I$. Ideja postopka izločanja s_i je, da scenarij s_i spremenimo tako, da postane osamljen. To naredimo tako, da v množici A_i ohranimo le en atribut, vse ostale pa odstranimo. V primeru ohranitve atributa a_{ik} bomo takemu postopku rekli *fiksiranje* atributa. Ohranjeni atribut na ta način postane osamljen, zato lahko $i \in V$ izločimo iz G na zgoraj opisan način. S fiksiranjem atributov $a_{ik} \in A_i$ dobimo $|A_i|$ različnih nalog. Najcenejša rešitev dobljenih nalog je hkrati tudi optimalna rešitev originalne naloge (G, I, C).

Izločanje scenarija

S pomočjo fiksiranja atributov in izločanja scenarijev z osamljenimi atributi lahko sestavimo algoritem za izločanje poljubnega scenarija. Za izvedbo postopka izločanja poljubnega scenarija $s_i \in I$ iz naloge (G, I, C) je torej potrebno za vsak $1 \leq k \leq |A_i|$ narediti naslednje:

1. iz naloge (G, I, C) s fiksiranjem atributa $a_{ik} \in A_i$ najprej konstruiramo nalogo (G, I_{ik}, C), kjer je $s_i \in I_k$ opisan z $A'_i = \{a_{ik}\}$;

2. iz naloge (G, I_{ik}, C) z izločanjem i iz G konstruiramo naloge $(G_{ij}, I_{ij}, C_{ijk})$, kjer $G_{ij} \in \mathcal{G}(i)$;

3. za vse $G_{ij} \in \mathcal{G}(i)$ poiščemo optimalno rešitev naloge $(G_{ij}, I_{ij}, C_{ijk})$;

4. iz optimalnih rešitev nalog $(G_{ij}, I_{ij}, C_{ijk})$, kjer $G_{ij} \in \mathcal{G}(i)$, sestavimo optimalno rešitev naloge (G, I_{ik}, C);

5. optimalna rešitev naloge (G, I, C) je najcenejša izmed vseh optimalnih rešitev nalog (G, I_{ik}, C), kjer $1 \leq k \leq |A_i|$.

V nalogah $(G_{ij}, I_{ij}, C_{ijk})$, kjer $G_{ij} \in \mathcal{G}(i)$ in $1 \leq k \leq |A_i|$, sta G_{ij} in I_{ij} odvisna le od i in j ter neodvisna od a_{ik}, medtem ko je C_{ijk} odvisna od vseh. V algoritmu za izločanje lahko to izkoristimo tako, da pravzaprav izločanje i delno opravimo, t.j. izračunamo G_{ij} in I_{ij}, že pred fiksiranjem atributov $a_{ik} \in A_i$. Ko konstruiramo naloge $(G_{ij}, I_{ij}, C_{ijk})$, torej opravljamo samo konstrukcijo C_{ijk}.

4.5.3 Algoritem

Lotimo se sedaj opisa algoritma za reševanje problemov prilagodljivih $p = 1$ atributov na splošnem grafu prehodov scenarijev. V tretjem koraku zgoraj opisanega postopka izločanja scenarija je potrebno poiskati optimalno rešitev naloge $(G_{ij}, I_{ij}, C_{ijk})$, kar je v primeru splošnega grafa G_{ij} še vedno $\mathcal{NP}$-težek problem. V primeru, da je G_{ij} drevo, pa imamo na voljo polinomski natančni algoritem.

Zato pri izbiri vozlišča $i \in V$ za izločitev skušamo izbrati tak i, katerega posledica izločitve bi bila, da so vsi grafi G_{ij}, kjer $j \in \mathcal{N}(i)$, drevesa. Žal to ni vedno mogoče. V takem primeru lahko na nalogah G_{ij}, I_{ij}, C_{ijk}, v katerih G_{ij} ni drevo, postopek izločanja ponovimo na nekem scenariju iz I_{ij}. Na ta način pridemo do naslednjega algoritma.

Algoritem 4.12: Izločanje scenarijev

Vhod: Naloga prilagodljivih atributov (G, I, C).
Izhod: Rešitev $(a_1, a_2, \ldots, a_n)$.

```
1.  procedure IzločanjeScenarijev;
2.  if drevo(G) then
3.     return Reši (G, I, C);
4.  Izberi i ∈ V;
5.  Konstruiraj 𝒢(i) z izločanjem i;
6.  forall a_ik ∈ A_i do
7.     Konstruiraj (G, I_ik, C) s fiksiranjem a_ik;
8.     forall G_ij ∈ 𝒢(i) do
9.        if |I_ij| = 1 then
10.          a_j := arg min_{a_jl ∈ A_j} c_ij(a_ik, a_jl);
11.       else
12.          Konstruiraj C_ijk;
13.          S_ijk := IzločanjeScenarijev((G_ij, I_ij, C_ijk));
14.          Uporabi S_ijk za sestavo S_ik = (a_1, a_2, ..., a_n);
15.       end;
16.    end;
17. end;
18. return arg min_{1≤k≤|A_i|} f_sum(S_ik);
```

Gre za rekurzivni algoritem, katerega ustavitveni pogoj (vrstica 2) je izpolnjen, kadar je G drevo. V tem primeru se naloga (G, I, C) optimalno reši (vrstica 3) z uporabo algoritma 4.8, sicer pa se izvede preostali del algoritma. V vrstici 4 se izbere vozlišče i za izločanje, nato (vrstica 5) konstruiramo $\mathcal{G}(i)$, kar lahko storimo še pred fiksiranjem atributov. Zatem (vrstici 6 in 7) za fiksiranje po vrsti izbiramo atribute $a_{ik} \in A_i$. V vrsticah od 8 do 16 poteka reševanje nalog $(G_{ij}, I_{ij}, C_{ijk})$, kjer $G_{ij} \in \mathcal{G}(i)$. Najprej (vrstica 9) preverimo, če naloga vsebuje le en scenarij, in v tem primeru (vrstica 10) izberemo atribut za scenarij s_j. Sicer (vrstica 12) pa konstruiramo

C_{ijk} in rekurzivno kličemo (vrstica 13) IzločanjeScenarijev na nalogi $(G_{ij}, I_{ij}, C_{ijk})$, katere rešitev uporabimo (vrstica 14) za sestavo rešitve S_{ik} naloge (G, I_{ik}, C). Na koncu (vrstica 18) vrnemo najcenejšo izmed rešitev S_{ik}, kjer $1 \leq k \leq |A_i|$.

Algoritem 4.12 je namenjen za izločanje scenarijev za primer sum prilagodljivosti, vendar ga lahko preprosto spremenimo, da je uporaben tudi za max prilagodljivost. To naredimo tako, da v zadnji vrstici f_{sum} nadomestimo z f_{max}.

Časovna zahtevnost

Za preverjanje ali je G drevo se porabi $O(m)$ časa. Prav toliko časa se porabi tudi za konstrukcijo $\mathcal{G}(i)$, ker je potrebno za vsako povezavo ugotoviti h kateremu podgrafu G_{ij} sodi. Zanka fiksiranja atributov $a_{ik} \in A_i$ se izvede $|A_i|$-krat, t.j. $O(r)$. Samo fiksiranje atributa oz. konstrukcijo (G, I_{ik}, C) lahko opravimo v konstantnem času, ker si je potrebno le zapomniti fiksirani atribut. Število različnih grafov G_{ij} je enako $|\mathcal{G}(i)| = O(\delta(i))$. Izbira a_j (vrstica 10) zahteva $O(r)$ časa. Podatke za C_{ijk} lahko beremo neposredno iz C. Rekurzivni klic zahteva $T(|I_{ij}|)$ časa, kjer je $T(n)$ časovna zahtevost algoritma. Ceno $f_{sum}(S_{ik})$ lahko izračunamo iz $f_{sum}(S_{ijk})$ tekom izvajanja zanke iz vrstice 8. Najcenejšo S_{ik}, kjer $1 \leq k \leq |A_i|$, pa lahko poiščemo tekom izvajanja zanke iz vrstice 6.

Vrstici 2 in 3 v algoritmu sta namenjeni za pohitritev algoritma v primeru, če je G drevo. Če ju iz algoritma izločimo, potem časovno zahtevnost takega algoritma opisuje naslednja rekurzivna enačba.

$$T(n) = |A_i| \sum_{G_{ij} \in \mathcal{G}(i)} T(|I_{ij}|) + O(n^2), \tag{4.15}$$

kjer je $i \in V$ vozlišče izbrano za izločanje in $O(n^2)$ časovna zahtevnost konstrukcije $\mathcal{G}(i)$. Na vsakem koraku rekurzije izločimo vsaj eno vozlišče, tako je globina rekurzije kvečjemu $O(n)$.

Časovna zahtevnost za polni graf. Oglejmo si kako je s časovno zahtevnostjo algoritma, če je G polni graf. V tem primeru ob vsakem klicu IzločanjeScenarijev izločimo vozlišče $i \in V$, tako da preostale naloge niso drevesa, razen ko preostaneta le še dve vozlišči. Pri tem velja $T(2) = O(r^2)$. Polni graf tako predstavlja najslabši primer za časovno zahtevnost algoritma, ki jo lahko v tem primeru zapišemo z naslednjo rekurzivno enačbo.

$$T(n, r) = r\,T(n-1, r) + O(n^2). \tag{4.16}$$

Že na prvi pogled je očitno, da gre v enačbi (4.16) za eksponentno odvisnost. Lahko pa po metodi rekurzijskega drevesa [CLRS01] ugotovimo natančnejšo zgornjo mejo.

$$\begin{aligned} T(n,r) &= c\sum_{i=0}^{n-3} r^i(n-i)^2 + r^{n-2}O(r^2) \leq \\ &\leq c\sum_{i=0}^{n-3} r^i n^2 + O(r^n) = \\ &= cn^2\frac{r^{n-2}-1}{r-1} + O(r^n) = \\ &= O(n^2 r^n), \end{aligned}$$

kjer je $c > 0$ neka konstanta. Pravilnost zadnje ugotovitve preverimo tako, da jo vstavimo v enačbo 4.16. Velja

$$\begin{aligned} T(n,r) &\leq rc_1(n-1)^2 r^{n-1} + c_2 n^2 = \\ &= c_1(n^2-2n+1)r^n + c_2 n^2 = \\ &= O(n^2 r^n). \end{aligned}$$

Algoritem je v najslabšem primeru torej superpolinomski, vseeno pa delitev grafa G na več manjših delov daje slutiti, da bi algoritem lahko bil precej hitrejši v nekaterih primeri, kot so npr. redki oz. slabo povezani grafi.

Časovna zahtevnost v povprečnem primeru. Skušajmo podati analizo časovne zahtevnost algoritma v povprečnem primeru. Pri tem naša analiza ne bo temeljila na verjetnostnih porazdelitvah vhodnih parametrov, ampak na njihovih povprečnih vrednostih. Naj bo d povprečno število delov $G_{ij} \in \mathcal{G}(i)$ na katere razpade G ob izločitvi poljubnega $i \in V$, t.j.

$$d = \frac{1}{n}\sum_{i \in V} |\mathcal{G}(i)|.$$

Velja $d \geq 1$. Za vsak $i \in V$ velja tudi

$$\sum_{G_{ij} \in \mathcal{G}(i)} |V_{ij}| = n - 1,$$

ker izločimo eno vozlišče, vsa ostala vozlišča pa pripadajo nekemu $G_{ij} \in \mathcal{G}(i)$. Izračunamo lahko torej povprečno velikost grafa G_{ij}, ki je

$$\overline{|V_{ij}|} = \frac{\sum_{i \in V}\sum_{G_{ij} \in \mathcal{G}(i)} |V_{ij}|}{\sum_{i \in V} |\mathcal{G}(i)|} = \frac{n-1}{d}.$$

Naj bo $r = 1/n \sum_{i \in V} |A_i|$, t.j. povprečno število atributov. Torej lahko zapišemo rekurzivno enačbo za povprečno zahtevnost algoritma

$$\overline{T}(n) = r\, d\, \overline{T}(\frac{n-1}{d}) + O(n^2).$$

Velja $T(n) = O(T'(n))$, kjer je

$$T'(n) = r\, d\, T'(\frac{n}{d}) + O(n^2).$$

Po metodi rekurzijskega drevesa zopet poiščimo zgornjo mejo za $T'(n)$. Velja

$$\begin{aligned} T'(n) &= cn^2 \sum_{i=0}^{\log_d n - 1} (\frac{r}{d})^i + O((rd)^{\log_d n} T(1)) = \\ &= c\frac{d}{r-d}(n^{\log_d r+1} - n^2) + O(n^{\log_d r+1} T(1)) = \\ &= O(n^{\log_d r+1} T(1)). \end{aligned}$$

Kar lahko dokažemo z metodo substitucije [CLRS01].

$$\begin{aligned} T'(n) &= r\, d\, T'(n/d) + O(n^2) \leq \\ &\leq rdc_1 T(1)(\frac{n}{d})^{\log_d r+1} + c_2 n^2 = \\ &= rdc_1 T(1)\frac{n^{\log_d r+1}}{rd} + c_2 n^2 \leq \\ &\leq c_1 T(1) n^{\log_d r+1}, \end{aligned}$$

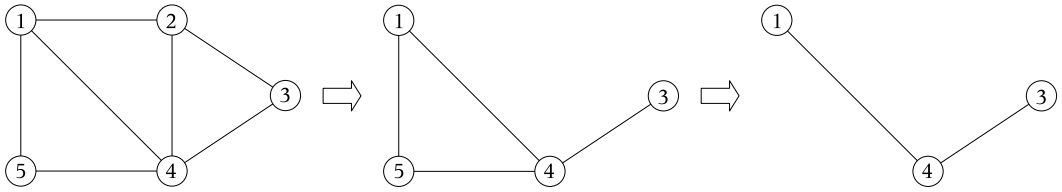
kjer je $c_1 > c_2$ in $r > d$.

4.5.4 Zahtevnost pretvarjanja grafa v drevo

V algoritmu 4.12 izberemo poljubno vozlišče za izločanje. Omenili pa smo že, da so pravzaprav boljše takšne izbire, ki bi kar najhitreje pripeljale do razbitja nastalih podgrafov v drevesa. Problem, ki se poraja, je pretvarjanje grafa v drevo. Natančneje, vprašanje je, kako iz grafa odstraniti neko podmnožico vozlišč, da bo preostanek grafa drevo oz. gozd. Pri tem je seveda zaželjeno, da je odstranjenih vozlišč kar se da malo. Na prvi pogled se problem zdi preprost, vendar bomo pokazali, da gre pravzaprav za $\mathcal{NP}$-težek problem.

Primer. Oglejmo si primer odstranjevanja vozlišč, prikazan na sliki 4.11. Na levi strani slike je prikazan graf s petimi vozlišči. Najprej iz njega odstranimo vozlišče 2 in dobimo graf na sredini slike. Nato iz tega odstranimo še vozlišče 5 in dobimo graf na desni strani slike, ki pa je drevo. Tako smo iz originalnega grafa v dveh korakih

Slika 4.11: Izločanje vozlišč.



oz. z dvema odstranjenima vozliščema dobili drevo. Optimalna rešitev pa je samo izločitev vozlišča 4.

Kot smo že povedali, v nadaljevanju za naše potrebe ni nujno, da je končni graf drevo, ampak zadostuje že, da je gozd. Problem pretvarjanja grafa v gozd je opisan z naslednjo definicijo.

Problem 4.17: Najmanjša Sprememba Grafa v Gozd

Naloga: Graf $G = (V, E)$.
Dopustna rešitev: Podmnožica $U \subseteq V$, tako da je graf $H(V/U, F)$ induciran nad G z vozlišči V/U gozd.
Kakovost dopustne rešitve: $|U|$.
Cilj: Minimizacija.

Da pokažemo $\mathcal{NP}$-težkost problema, potrebujemo nek $\mathcal{NP}$-poln problem. Naj bo $G = (V, E)$ graf. Množica $U \subseteq V$, ki vsebuje vsaj eno vozlišče vsake povezave $(u, v) \in E$ se imenuje *vozliščno pokritje* grafa G. Iskanje vozliščnega pokritja moči kvečjemu B je $\mathcal{NP}$-poln problem [GJ79]. Definirajmo ga.

Problem 4.18: Vozliščno pokritje

Naloga: Neusmerjen graf $G = (V, E)$ in parameter $B \in \mathbb{Z}^+$.
Vprašanje: Ali obstaja vozliščno pokritje $C \subseteq V$ grafa G, tako da $|C| \leq B$?

Sedaj lahko zapišemo in dokažemo naslednjo trditev.

Trditev 4.33 ▶ Najmanjša Sprememba Grafa v Gozd *je $\mathcal{NP}$-težek optimizacijski problem.*

Dokaz. Označimo s P odločitveno različico problema Najmanjša Sprememba Grafa v Gozd. Očitno je $P \in \mathcal{NP}$. Preostali del dokaza poteka s prevedbo problema Vozliščno pokritje.

Iz naloge $G = (V, E)$ za problem Vozliščno pokritje ustvarimo nalogo $H = (V \cup V', E \cup E')$ za P. Definirajmo

$$V' = \{w_{uv} | (u, v) \in E\}$$

in

$$E' = \{(u, w_{uv}), (v, w_{uv}) | w_{uv} \in V\}.$$

Drugače povedano, za vsako povezavo $(u, v) \in E$ v grafu G smo v H ustvarili kliko na treh vozliščih u, v in w_{uv}. Pokazali bomo, da z razbitjem (odstranitvijo vozlišča) te klike pravzaprav pokrijemo povezavo $(u, v) \in E$. Natančneje, za nalogo G problema Vozliščno pokritje obstaja rešitev U, kjer $|U| \leq k$ natanko takrat, ko za nalogo H problema P obstaja rešitev U', kjer $|U'| \leq k$.

($\Rightarrow$) Naj bo U rešitev naloge G problema Vozliščno pokritje. Velja $G \subseteq H$. Potemtakem je v H potrebno razbiti cikle v G, poleg tega pa tudi cikle, ki smo jih v H pridali na povezave. Ker je U vozliščno pokritje v G, za vsak $(u, v) \in E$ velja $u \in U$ ali $v \in U$ (lahko tudi oboje).

Naj bo $c = (e_1, e_2, \ldots, e_l)$ poljuben cikel (če le-ta obstaja) v G. Za vsako povezavo $e_i = (u, v)$ cikla velja $u \in U$ ali $v \in V$. Cikel c je torej razbit. Nadalje naj bo $(u, v), (u, w_{uv}), (v, w_{uv})$ nek cikel v H, ki ni cikel v G. Ker velja $u \in U$ ali $v \in U$, je tudi ta cikel razbit. Rešitev naloge H je torej $U' = U$, in velja $|U'| = |U| \leq k$.

($\Leftarrow$) Naj bo U' rešitev naloge H problema P. Torej U' razbije vse cikle v H, in s tem tudi cikle $(u, v), (u, w_{uv}), (v, w_{uv})$ za $(u, v) \in E$. Potemtakem, za vsako povezavo $(u, v) \in E$ velja, da je v U' vsaj eno od vozlišč u, v, ali w_{uv}. Če je to u ali v, potem je povezava $(u, v) \in E$ pokrita, sicer pa $w_{uv} \in U'$ zamenjamo z enim od u ali v in tako dobimo vozliščno pokritje U.

Natančneje, definirajmo

$$U = U'/V' \cup \{u | w_{uv} \in U' \cap V'\}.$$

Prvi del enačbe pove, da v U' ohranimo vsa vozlišča v, za katera velja $v \notin V'$ (oz. $v \in V$). Drugi del enačbe pa pove, da v U' vsa vozlišča $w_{uv} \in U'$, za katera velja $v \in V'$, zamenjamo z $u \in V$[9]. Ker sta V in V' disjunktni množici, opisana zamenjava ohranja moč množice, t.j. $|U| = |U'| \leq k$. □

Problem Najmanjša Sprememba Grafa v Gozd je pravzaprav "lažja" različica problema Najmanjša Sprememba Grafa v Drevo, sledi torej, da je tudi slednji $\mathcal{NP}$-težek.

Oglejmo si še naslednji alternativni način, s katerim lahko pokažemo $\mathcal{NP}$-težkost obeh problemov. Definirajmo naslednji problem.

Problem 4.19: Največji Induciran Podgraf z Lastnostjo p

Naloga: Graf $G = (V, E)$ in lastnost p.
Dopustna rešitev: Podmnožica $U \subseteq V$ tako, da ima graf $H(V/U, F)$ induciran nad G z vozlišči V/U lastnost p.
Kakovost dopustne rešitve: $|V/U|$.
Cilj: Maksimizacija.

[9]Lahko bi izbrali tudi $v \in V$, vendar ta izbira za dokaz ni pomembna.

Problem[10] je večkrat definiran tudi kot minimizacija $|U|$. Opišimo naslednji dve značilnosti lastnosti p:

- *trivialnost* pomeni, da ne obstaja neskončno mnogo grafov z lastnostjo p in ravno tako ne obstaja neskončno grafov brez lastnosti p;
- *dednost* pomeni da, če lastnost p velja za graf G, potem velja tudi za vsak njegov podgraf $H \subseteq G$, t.j. če $p(G)$ potem tudi $p(H)$, kjer je $H \subseteq G$.

Problem Največji Induciran Podgraf z Lastnostjo p je $\mathcal{NP}$-težek, če je lastnost p dedna in ni trivialna [ACG+99]. Preprosto lahko ugotovimo, da obe značilnosti veljata tako za lastnost "graf je drevo" kot za "graf je gozd". Netrivialnost velja, ker obstaja neskončno mnogo grafov, ki so drevesa (gozd), kakor tudi neskončno grafov, ki niso drevesa (gozd). Dednost pa, ker je vsak podgraf drevesa (gozda) tudi drevo (gozd).

Pametno izbiranje vozlišč za izločanje v algoritmu 4.12 je torej $\mathcal{NP}$-težek problem. Kljub temu je verjetno smiselno v algoritem vpeljati neko hevristiko za izbiro vozlišče, npr. izbira $i \in V$ z največjo stopnjo $\delta(i)$. Kakšen graf pa je najslabši primer grafa za spreminjanje v drevo? Največ vozlišč, da dobimo drevo, je potrebno izločiti iz polnega grafa. Izločiti je potrebno natanko $n - 2$ vozlišč.

4.6 Dvostopenjski problemi investiranja

4.6.1 Ogrodje dvostopenjskih problemov investiranja

V družino *dvostopenjskih problemov* sodijo problemi, v katerih je prisotna tudi časovna komponenta. V teh problemih nastopata, kot pove že samo ime, dva *časovna trenutka*, t.j. znana *sedanjost* in neznana oz. negotova *prihodnost*. V sedanjosti se za znano situacijo oz. scenarij sprejme neka odločitev; ker pa prihodnost ni znana, v sedanjosti sprejeta odločitev morda ni najbolj primerna za dejansko realizirani scenarij v prihodnosti. Vendar je v prihodnosti možno poprej sprejeto odločitev popraviti. Seveda je za popravek že sprejete odločitve potrebno plačati neko ceno. Tej ceni pravimo tudi *nadomestilo* ali *odškodnina*[11]. Cilj optimizacije je minimizirati plačano odškodnino.

Prevedimo opisano na terminologijo optimizacije scenarijev. Vhodni podatki problema so scenariji, od katerih en scenarij predstavlja sedanjost, ostali pa prihodnost. Graf prehodov scenarijev vsebuje povezave od sedanjega scenarija k prihodnjim. Cilj je poiskati optimalno prilagodljive partikularne rešitve za vse scenarije.

V nadaljevanju se bomo ukvarjali s posebno vrsto dvostopenjskih problemov, ki jih imenujemo *dvostopenjski problemi investiranja* [MMRR04]. Gre za dvostopenjske probleme, v katerih so partikularne rešitve za prihodnje scenarije poznane

[10] V [ACG+99] je problem poimenovan Minimum Node Deletition to Obtain Graph with Property p.

[11] *angl. recourse cost*

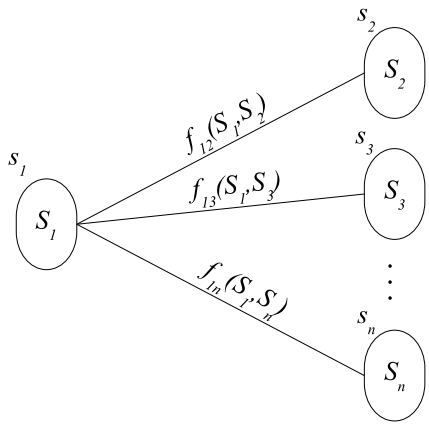
oz. podane, potrebno je poiskati le rešitev za sedanji scenarij. To pomeni, da je potrebno poiskati le rešitev S_1 za sedanji scenarij s_1, ostale rešitve $S_2, S_3, \ldots, S_n$ prihodnjih scenarijev pa so podane kot vhod problema.

Lahko se vprašamo, ali je v praksi smiselno, da so rešitve za prihodnje scenarije podane. Recimo, da scenariji predstavljajo pričakovane situacije razvoja nekega podjetja. Podjetje pravzaprav v sedanjosti ne ve, kateri od scenarijev se bo uresničil, vendar v primeru uresničitve poljubnega scenarija dobro ve, kako bo reagiralo. Torej pozna rešitve za morebitne pričakovane scenarije. Vendar si podjetje že v sedanjosti želi nastopati na trgu in hkrati želi biti pripravljeno na vse prihodnje scenarije, zato želi že v sedanjosti poiskati neko rešitev, ki bo kar najbolj prilagodljiva glede na pričakovane scenarije. V nekem smislu podjetje želi že v sedanjosti investirati v (negotovo) prihodnost.

Taki in podobni dvostopenjski problemi se pojavljajo na področju operacijskih raziskav, znanosti ravnateljevanja oz. upravljanja[12], sprejemanja odločitev[13] itd. [Hen99, RS03, SS, Sny04]. Načini za njihovo reševanje največkrat sodijo v področje stohastičnega programiranja. Mi jih bomo v nadaljevanju obdelali predvsem z vidika problemov prilagodljivih atributov.

Ogrodje dvostopenjskih problemov prilagodljivih atributov je prikazano na sliki 4.12. V problemu nastopa n scenarijev, od katerih scenarij s_1 predstavlja znano

Slika 4.12: Ogrodje dvostopenjskih problemov prilagodljivih atributov.



sedanjo situacijo, ostali scenariji $s_2, s_3, \ldots, s_n$ pa predstavljajo neznano prihodnost. Scenarij s_i je opisan z množico atributov A_i. Dopustna partikularna rešitev za s_i je $S_i \subseteq A_i$, kjer $|S_i| = p$. Funkcija $f_{ij}(S_i, S_j)$ predstavlja ceno transformacije S_i v S_j.

V problemih investiranja dodatno velja, da so $S_2, S_3, \ldots, S_n$ podani kot vhod problema, za $2 \leq i \leq n$ torej velja $S_i = A_i$. Celotne rešitve so torej oblike $(S_1, A_2, A_3, \ldots, A_n)$, kjer $S_1 \subseteq A_1$ in $|S_1| = p$. Za nas je pravzaprav zanimiv samo S_1, zato bomo v nadaljevanju namesto $(S_1, A_2, A_3, \ldots, A_n)$ večkrat pisali samo S_1.

[12] *angl. management science*
[13] *angl. decision making*

Potemtakem gre za probleme prilagodljivih atributov, v katerih:

- je graf prehodov scenarijev dvostopenjski; in
- velja $S_i = A_i$, kjer $|S_i| = p$, za vsak $i \geq 2$.

V nadaljevanju si bomo te probleme najprej ogledali z vidika sum prilagodljivosti, nato še z vidika max prilagodljivosti. Tako bomo obdelali naslednja dva problema:

- Najmanjše 2-Stopenjsko $\sum$-Prilagodljivo Investiranje; in
- Najmanjše 2-Stopenjsko max-Prilagodljivo Investiranje.

4.6.2 Sum prilagodljivost

V tem podrazdelku se bomo lotili dvostopenjskih problemov investiranja z vidika sum prilagodljivosti. Glede na število scenarijev, ki nastopajo v problemu, bomo pokazali zahtevnost reševanja. Tako bomo predstavili polinomski natančni algoritem za $n = 3$ scenarije ter pokazali $\mathcal{NP}$-težkost in neaproksimabilnost za $n \geq 4$ scenarije.

Natančno reševanje za $n = 3$ scenarije

Predstavili bomo polinomski algoritem za optimalno reševanje problema Najmanjše 2-Stopenjsko $\sum$-Prilagodljivo Investiranje, v katerem bo število scenarijev enako $n = 3$. Ker gre za dvostopenjski graf prehodov scenarijev, to pomeni, da v problemu nastopa en sedanji scenarij in dva prihodnja, za katera sta že podani partikularni rešitvi. Pokazali bomo, kako lahko v tem primeru sestavimo algoritem s pomočjo problema Najmanjša Cena Pretoka.

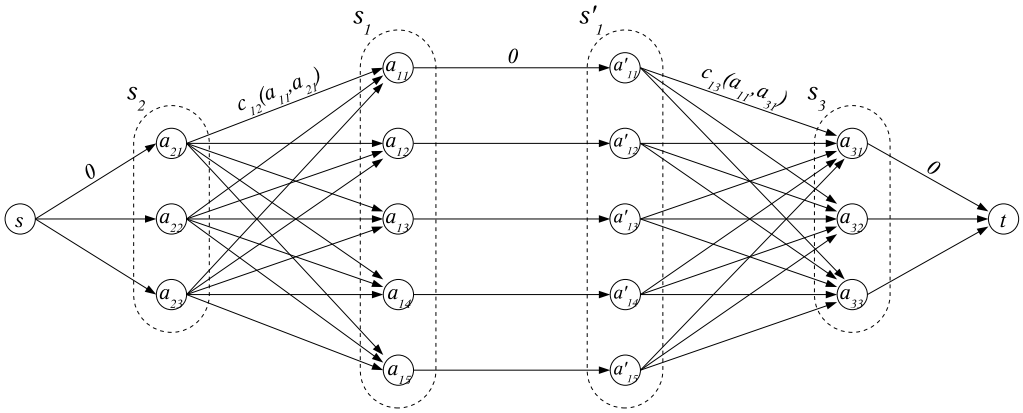
Spomnimo, da gre v Najmanjše 2-Stopenjsko $\sum$-Prilagodljivo Investiranje pravzaprav za problem Najmanjša $\sum$-Prilagodljivost p-Atributov na dvostopenjskem grafu prehodov, v katerem poleg tega velja še $S_i = A_i$ in $p = |S_i| = |A_i|$. Zapišimo naslednjo trditev, na kateri bo temeljil algoritem.

Trditev 4.34 ▶ *Problem* Najmanjše 2-Stopenjsko $\sum$-Prilagodljivo Investiranje*, v katerem nastopajo trije ($n = 3$) scenariji, je optimalno rešljiv v polinomskem času.*

Dokaz. Iz naloge (G, I, C, p) za Najmanjše 2-Stopenjsko $\sum$-Prilagodljivo Investiranje bomo ustvarili graf $H = (U, F)$ za problem Najmanjša Cena Pretoka. Velja torej, $G = (V, E)$ je dvostopenjski graf prehodov scenarijev, za katerega velja $|V| = |I| = 3$. Naj bo s_1 scenarij, ki predstavlja sedanjost, s_2 in s_3 pa naj predstavljata prihodnost. Konstruirajmo omrežje H za problem pretokov na naslednji način.

Scenarij s_1 (s tem tudi množico A_1) podvojimo in tako dobimo scenarij s'_1 (A'_1). Nato vse atribute vseh scenarijev pretvorimo v vozlišča U grafa H, v katerega dodamo tudi vozlišči s in t. Graf H je tako sestavljen iz vozlišč $U = A_1 \cup A'_1 \cup A_2 \cup A_3 \cup \{s, t\}$. Skupno število vozlišč v H je torej $2(|A_1|+p+1) = O(|A_1|+p)$. Primer konstrukcije grafa H je prikazan na sliki 4.13. (Za na sliki prikazano konstrukcijo velja $|A_1| = 5, |A_2| = |A_3| = p = 3$.)

Slika 4.13: Pretokovni model sum investiranja.



Dodajmo v H še povezave. Najprej dodamo povezave s ceno 0 in kapaciteto 1, t.j. povezave med izvorom s do vseh vozlišč iz A_2, iz vozlišč iz A_3 v ponor t in povezave iz $a_{1k} \in A_1$ v ustrezna podvojena $a'_{1k} \in A'_1$. Natančneje, za vse $a_{2k} \in A_2$ velja $(s, a_{2k}) \in F$, nadalje za vse $a_{3k} \in A_3$ velja $(a_{3k}, t) \in F$ in še za vse $a_{1k} \in A_1$ ter $a'_{1k} \in A'_1$ velja $(a_{1k}, a'_{1k}) \in F$.

Nato dodamo povezave s ceno $c_{12}(a_{1k}, a_{2l})$ in kapaciteto 1 med vozlišči iz A_2 in A_1. Natančneje, za vse $a_{2l} \in A_2$ in $a_{1k} \in A_1$ dodamo povezavo $(a_{2l}, a_{1k}) \in F$. Poleg tega dodamo še povezave s ceno $c_{13}(a_{1k}, a_{3l})$ in kapaciteto 1 med vozlišči A'_1 in A_3. Natančneje, za vse $a'_{1k} \in A_1$ in $a_{3l} \in A_3$ velja $(a'_{1k}, a_{3l}) \in F$. Število vseh povezav, ki jih tako dodamo v H, je enako $2p + |A_1| + 2p|A_1| = O(p|A_1|)$.

V nadaljevanju dokaza bomo pokazali, da je cena rešitve S_1 naloge (G, I, C, p) problema Najmanjše 2-Stopenjsko $\sum$-Prilagodljivo Investiranje enaka ceni ustrezne rešitve (f_{uv}) naloge H problema Najmanjša Cena Pretoka in obratno.

($\Rightarrow$) Naj bo S_1 rešitev naloge (G, I, C, p) problema Najmanjše 2-Stopenjsko $\sum$-Prilagodljivo Investiranje. Sestavimo rešitev (f_{uv}) za nalogo H problema Najmanjša Cena Pretoka. Najprej v H ustvarimo p disjunktnih poti iz s v t, ki potekajo skozi vozlišča iz množic A_2, S_1, S'_1 in A_3, kjer je S'_1 podvojena S_1.

Dvoumnosti izbire povezav iz A_2 v S_1 in iz S_1 v A_3 rešujemo s pomočjo problema Najmanjša Dodelitev[14]. Natančneje, med A_2 in S_1 izberimo povezave, ki so v

[14] Algoritem za problem Najmanjša Dodelitev temelji na algoritmu za problemu Najmanjša Cena Pretoka, kar omogoča, da rešitev prvega neposredno ustreza rešitvi drugega.

rešitvi problema Najmanjša Dodelitev na nalogi $H_{12} = (S_1 \cup A_2, S_1 \times A_2)$, in med S_1' in A_3 izberemo povezave v rešitvi naloge $H_{13} = (S_1' \cup A_3, S_1' \times A_3)$.

Pretok f_{uv} skozi povezavo $(u, v) \in F$ naj bo enak 1, če (u, v) pripada kateri izmed disjunktnih poti iz s v t; sicer naj bo enak 0. Pretok skozi tako ustvarjene disjunktne poti ima prav gotovo vrednost p. Izračunajmo še ceno pretoka (f_{uv}).

$$\begin{aligned}
cost((f_{uv})) &= \sum_{(u,v)\in F} f_{uv}w_{uv} = \\
&= \sum_{u\in A_2, v\in S_1} f_{uv}w_{uv} + \sum_{u\in S_1', v\in A_3} f_{uv}w_{uv} = \\
&= \sum_{(u,v)\in M_{12}} w_{uv} + \sum_{(u,v)\in M_{13}} w_{uv} = \\
&= \sum_{(u,v)\in M_{12}} c_{12}(u, v) + \sum_{(u,v)\in M_{13}} c_{13}(u, v) = \\
&= f_{12}(S_1, A_2) + f_{13}(S_1, A_3) = \\
&= f_{sum}(S_1, A_2, A_3)
\end{aligned}$$

Pri tem sta M_{12} in M_{13} rešitvi problema Najmanjša Dodelitev na nalogah H_{12} in H_{13}.

($\Leftarrow$) Naj bo (f_{uv}) rešitev s pretokom vrednosti p naloge $H = (U, F)$ problema Najmanjša Cena Pretoka. Trditev 2.5 nam zagotavlja, da so vrednosti f_{uv}, kjer $(u, v) \in F$, celoštevilske. To pomeni, da pretok pravzaprav teče po p disjunktnih poteh iz s v t. Naj bo $S_1 \subseteq A_1$ množica vozlišč, skozi katera gredo te poti. Velja $|S_1| = p$. Množica S_1 je prav gotovo dopustna rešitev problema Najmanjše 2-Stopenjsko $\sum$-Prilagodljivo Investiranje. Enako kot zgoraj velja $cost((f_{uv})) = f_{sum}(S_1, A_2, A_3)$.

Sledi torej, da če je cena pretoka (f_{uv}) najmanjša, potem je tudi celotna cena prilagajanja $f_{sum}(S_1, A_2, A_3)$ najmanjša. Če to ne bi bilo res, bi obstajal nek $S' \subseteq A_1$, tako da bi $f_{sum}(S', A_2, A_3) < f_{sum}(S_1, A_2, A_3)$. To pa bi pomenilo, da obstaja nek pretok (f'_{uv}), za katerega $cost((f'_{uv})) < cost((f_{uv}))$, s čimer pridemo do protislovja. Ker za problem Najmanjša Cena Pretoka obstajajo polinomski optimalni algoritmi, je trditev tako dokazana. □

Opis konstrukcije v dokazu nam omogoča, da ustvarimo polinomski algoritem za optimalno reševanje problema Najmanjše 2-Stopenjsko $\sum$-Prilagodljivo Investiranje, v katerem nastopajo trije scenariji. Algoritem je formalno podan v naslednji obliki.

Algoritem 4.13: Dvostopenjsko investiranje za $n = 3$.

Vhod: Naloga (G, I, C, p) prilagodljivih atributov, kjer je G dvostopenjski graf.
Izhod: Rešitev S_1.
1. Iz naloge (G, I, C, p) sestavi omrežje $H = (U, F)$;
2. V H poišči najcenejši pretok (f_{uv}) vrednosti p;
3. Sestavi rešitev S_1;

V prvi vrstici algoritma gre za konstrukcijo iz dokaza trditve 4.34, njena časovna zahtevnost je sicer $O(|A_1|p)$, vendar lahko podatke omrežja H neposredno dobimo iz (G, I, C, p) v konstantnem času; tako je prvi korak lahko vključen v algoritem za iskanje najcenejšega pretoka, ki ga lahko rešimo v času $O(|U|^3) = O((|A_1| + p)^3)$. Sestavljanje rešitve S_1 lahko izvedemo v $O(|A_1|)$ korakih. Celotna časovna zahtevnost algoritma je torej $O((p + r)^3)$.

Zahtevnost reševanja za $n \geq 4$ scenarije

Vprašajmo se, kako je z reševanjem v primeru, kadar graf prehodov scenarijev vsebuje štiri ali morda še več scenarijev. Pokazali bomo, da je v tem primeru problem Najmanjše 2-Stopenjsko $\sum$-Prilagodljivo Investiranje $\mathcal{NP}$-težek. To bomo storili s prevedbo problema 3D-Ujemanje, ki je znan $\mathcal{NP}$-poln problem [GJ79]. Najprej naštejmo nekaj definicij.

Definicija 4.2: 3D-ujemanje

Naj bo $T \subseteq A \times B \times C$, pri čemer so množice A, B in C paroma disjunktne. Podmnožica $T' \subseteq T$ se imenuje *3D-ujemanje*, če se nobeno od mest (koordinat) v poljubnih dveh trojicah v T' ne ujema.

3D-ujemanje $T' \subseteq T$ torej predstavlja nekakšne neodvisne trojice. V povezavi s tem definirajmo naslednji odločitveni problem.

Problem 4.20: 3D-Ujemanje

Naloga: Množica $T \subseteq A \times B \times C$, pri čemer so A, B in C paroma disjunktne. Velja $|A| = |B| = |C| = p$.
Vprašanje: Ali obstaja 3D-ujemanje $T' \subseteq T$, kjer velja $|T'| = p$?

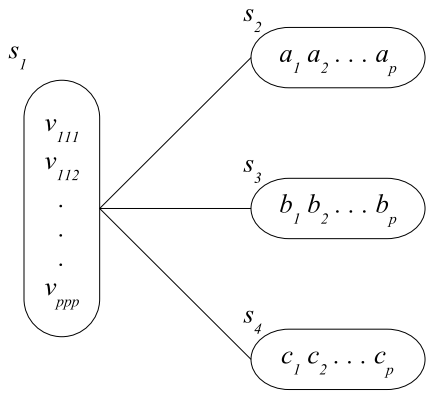
Zapišimo in dokažimo naslednji izrek.

Trditev 4.35 ▶ *Odločitvena različica problema* Najmanjše 2-Stopenjsko $\sum$-Prilagodljivo Investiranje, *v katerem nastopajo štirje ($n = 4$) scenariji, je $\mathcal{NP}$-polna.*

Dokaz. Iz naloge T, kjer so $A = \{a_1, a_2, \ldots, a_p\}$, $B = \{b_1, b_2, \ldots, b_p\}$ in $C = \{c_1, c_2, \ldots, c_p\}$, za problem 3D-Ujemanje bomo sestavili nalogo (G, I, C, p) za Najmanjše 2-Stopenjsko $\sum$-Prilagodljivo Investiranje, v katerem je G dvostopenjski graf, ki vsebuje $n = 4$ scenarije. Označimo slednji problem s P. Problem P prav gotovo pripada razredu $\mathcal{NP}$, zato se lotimo konstrukcije.

Ustvarimo dvostopenjski graf prehodov scenarijev $G = (V, E)$ s štirimi scenariji, kot je prikazano na sliki 4.14. Scenarij s_1 je opisan z A_1, tako da je vsaka trojica $(a_i, b_j, c_k) \in T$ predstavljena z atributom $v_{ijk} \in A_1$. Scenariji s_2, s_3 in s_4 so opisani z množicami atributov $A_2 = A$, $A_3 = B$ in $A_4 = C$, kjer A_i opisuje scenarij s_i. Na ta način skupaj ustvarimo $|T| + 3p$ atributov.

Slika 4.14: Konstrukcija naloge.



Definirajmo še cene prilagajanja atributov. Najprej definirajmo $c_{12}(v_{ijk}, a_l)$, kjer je $v_{ijk} \in A_1$ in $a_l \in A_2$, kot

$$c_{12}(v_{ijk}, a_l) = \begin{cases} 0 & i = l \\ \infty & \text{sicer.} \end{cases}$$

Nato $c_{13}(v_{ijk}, b_l)$, kjer je $v_{ijk} \in A_1$ in $b_l \in A_3$, kot

$$c_{13}(v_{ijk}, b_l) = \begin{cases} 0 & j = l \\ \infty & \text{sicer.} \end{cases}$$

In še $c_{14}(v_{ijk}, c_l)$, kjer je $v_{ijk} \in A_1$ in $c_l \in A_4$, kot

$$c_{14}(v_{ijk}, c_l) = \begin{cases} 0 & k = l \\ \infty & \text{sicer.} \end{cases}$$

Izbira atributov v s_1 predstavlja izbiro trojic za 3D-Ujemanje. Množica $T' \subseteq T$, kjer $|T'| = p$, je rešitev za problem 3D-Ujemanje natanko takrat, kadar je trojicam T ustrezna množica $S_1 \subseteq A_1$ rešitev s ceno $f_{sum}(S_1) = 0$ problema P na nalogi, sestavljeni na zgornji način.

($\Rightarrow$) Naj bo $T' \subseteq T$, kjer $|T'| = p$, rešitev problema 3D-Ujemanje. Definirajmo $S_1 \subseteq A_1$ kot $S_1 = \{v_{ijk} \in A_1 | (a_i, b_j, c_k) \in T'\}$ in izračunajmo $f_{sum}(S_1)$. Velja

$$f_{sum}(S_1) = f_{12}(S_1, A_2) + f_{13}(S_1, A_3) + f_{14}(S_1, A_4).$$

Ker je T' 3D-ujemanje na T, za vsak $a_i \in A = A_2$ obstaja $(a_i, b_j, c_k) \in T'$. Potemtakem za $a_i \in A_2$ in v_{ijk} velja $c_{12}(v_{ijk}, a_i) = 0$ in posledično še $f_{12}(S_1, A_2) = 0$. Analogno sklepamo še za vse $b_j \in B$ in $c_k \in C$ ter tako ugotovimo še $f_{13}(S_1, A_3) = f_{14}(S_1, A_4) = 0$. Zaključimo $f_{sum}(S_1) = 0$.

($\Leftarrow$) Naj bo S_1, kjer $|S_1| = p$, rešitev problema P s ceno $f_{sum}(S_1) = 0$. Definirajmo $T' \subseteq T$ kot $T' = \{(a_i, b_j, c_k) \in T | v_{ijk} \in S_1\}$. Ker $|S_1| = p$ sledi $|T'| = p$ in ker $f_{sum}(S_1) = 0$, sledi še $f_{12}(S_1, A_2) = f_{13}(S_1, A_3) = f_{14}(S_1, A_4) = 0$. Posledično za vsak $a_l \in A_2$ obstaja $v_{ijk} \in S_1$, tako da $c_{12}(v_{ijk}, a_l) = 0$, kar po definiciji c_{12} pomeni, da $a_l = a_i$. Torej za vsak $a_i \in A$ obstaja $(a_i, b_j, c_k) \in T'$. Analogno sklepamo še za vse $b_l \in A_3$ in $c_l \in A_4$ in ugotovimo, da je T' 3D-ujemanje na T. □

Neposredno sledi naslednja trditev.

Trditev 4.36 ▶ Najmanjše 2-Stopenjsko $\sum$-Prilagodljivo Investiranje, *v katerem nastopajo štirje ali več ($n \geq 4$) scenariji, je $\mathcal{NP}$-težek optimizacijski problem.*

Neaproksimabilnost za $n \geq 4$ scenarije

Trditev 4.37 ▶ *Za* Najmanjše 2-Stopenjsko $\sum$-Prilagodljivo Investiranje, *v katerem nastopajo štirje ali več ($n \geq 4$) scenarijev, ne obstaja polinomski aproksimacijski algoritem s konstantnim aproksimacijskim faktorjem, razen če $\mathcal{P}$=$\mathcal{NP}$.*

Dokaz. Neaproksimabilnost Najmanjše 2-Stopenjsko $\sum$-Prilagodljivo Investiranje bomo dokazali s pomočjo konstrukcije iz dokaza trditve 4.35. V omenjenem dokazu zamenjamo definicijo c_{12} z naslednjo

$$c_{12}(v_{ijk}, a_l) = \begin{cases} 1 & i = l \\ 1 + 3p\rho & \text{sicer.} \end{cases}$$

Podobno naredimo še za $c_{13}(v_{ijk}, a_l)$ in $c_{12}(v_{ijk}, a_l)$.

Nato iz naloge x za 3D-Ujemanje ustvarimo nalogo x' za Najmanjše 2-Stopenjsko $\sum$-Prilagodljivo Investiranje. Rešitev naloge x' ima vrednost $3p$, če in samo če v x obstaja 3D-ujemanje. V tem primeru ima naslednja najmanjša približna rešitev vrednost $3p(1 + \rho)$. V nasprotnem primeru, če v x ne obstaja 3D-ujemanje, potem ima optimalna rešitev naloge x vrednost vsaj $3p(1 + \rho)$. Torej,

če bi imeli polinomski ρ-aproksimacijski algoritem A za problem NAJMANJŠE 2-STOPENJSKO $\sum$-PRILAGODLJIVO INVESTIRANJE, bi z njim lahko v polinomskem času reševali problem 3D-UJEMANJE na naslednji način. Algoritem A bi pognali na nalogi x', ki ustreza nalogi x za 3D-UJEMANJE, in odgovorili z DA, če in samo če bi A vrnil rešitev vrednosti $3p$. □

4.6.3 Max prilagodljivost

Zahtevnost reševanja za $n = 3$ scenarije

Lotimo se še dvostopenjskih problemov investiranja z vidika max prilagodljivosti. Najprej si oglejmo, kako je z zahtevnostjo reševanja problema NAJMANJŠE 2-STOPENJSKO max-PRILAGODLJIVO INVESTIRANJE, v katerem nastopajo trije scenariji. Pokazali bomo, da je ta problem $\mathcal{NP}$-težek. To bomo pokazali s prevedbo problema RAZDELITEV, ki je $\mathcal{NP}$-poln [GJ79], na odločitveno različico NAJMANJŠE 2-STOPENJSKO max-PRILAGODLJIVO INVESTIRANJE na treh scenarijih.

Problem RAZDELITEV je podan z naslednjo definicijo.

Problem 4.21: RAZDELITEV

Naloga: Množica elementov $A = \{a_1, a_2, \ldots, a_p\}$, kjer elementu $a_i \in A$ pripada utež $w_i \in \mathbb{Z}^+$. Velja $\sum_i w_i = 2B$.
Vprašanje: Ali obstaja razdelitev $X, Y \subseteq A$, tako da velja $\sum_{a_i \in X} w_i = \sum_{a_i \in Y} w_i = B$?

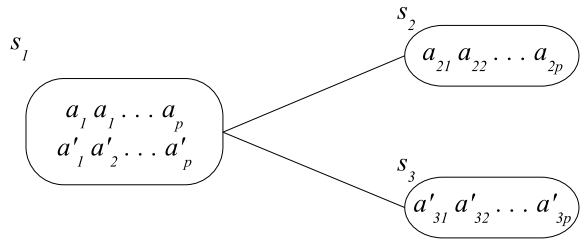
Dokažimo najprej naslednjo lemo.

Lema 4.38 ▶ *Odločitvena različica problema* NAJMANJŠE 2-STOPENJSKO max-PRILAGODLJIVO INVESTIRANJE*, v katerem nastopajo trije scenariji ($n = 3$), je $\mathcal{NP}$-polna.*

Dokaz. Označimo s P odločitveno različico problema NAJMANJŠE 2-STOPENJSKO max-PRILAGODLJIVO INVESTIRANJE, v katerem nastopajo trije scenariji. Ker je max-PRILAGODLJIVOST p-ATRIBUTOV v $\mathcal{NP}$, je v $\mathcal{NP}$ prav gotovo tudi P.

Iz naloge problema RAZDELITEV, t.j. množice A, uteži w_i in parametra B, bomo ustvarili nalogo (G, I, C, p) problema P. Ustvarimo graf prehodov scenarijev $G = (V, E)$, kot je prikazano na sliki 4.15. Definirajmo $A' = \{a'_1, a'_2, \ldots, a'_p\}$, kjer $a_i \in A$, torej A' vsebuje dvojnike elementov v A. Scenarij s_1 je opisan z množico $A_1 = A \cup A'$, s_2 je opisan z $A_2 = \{a_{21}, a_{22}, \ldots, a_{2p}\}$ in s_3 z $A_3 = \{a_{31}, a_{32}, \ldots, a_{3p}\}$. Ustvarili smo torej tri scenarije in $4p$ atributov. S tem je prevedba prav gotovo polinomska.

Slika 4.15: Konstrukcija naloge.



Potrebno je definirati še ceni c_{12} in c_{13} prilagajanja atributov. Najprej definirajmo $c_{12}(a, a_{2l})$, kjer $a \in A_1$ in $a_{2l} \in A_2$. Velja

$$c_{12}(a, a_{2l}) = \begin{cases} w_k & a = a_k \in A, k = l \\ 0 & a = a'_k \in A', k = l \\ 2B + 1 & \text{sicer.} \end{cases}$$

Nato definirajmo še $c_{13}(a, a_{3l})$, kjer je $a \in A_1$ in $a_{3l} \in A_3$, ki je

$$c_{13}(a, a_{3l}) = \begin{cases} 0 & a = a_k \in A, k = l \\ w_k & a = a'_k \in A', k = l \\ 2B + 1 & \text{sicer.} \end{cases}$$

S tem je prevedba definirana, potrebno je pokazati še njeno pravilnost. Za nalogo problema Razdelitev obstaja rešitev $X, Y \subseteq A$ natanko takrat, ko za ustrezno nalogo (G, I, C, p) problema P obstaja rešitev $S_1 = X \cup Y'$ s ceno kvečjemu B, pri čemer so v $Y' \subseteq A'$ dvojniki elementov iz Y.

($\Leftarrow$) Naj bo S_1 rešitev naloge problema P, kjer $f_{max}(S_1) \leq B$. Ker $|S_1| = p$, za vsak $a_i \in A$ in $a'_i \in A'$ velja, da bodisi $a_i \in S_1$ bodisi $a'_i \in S_1$, t.j. a_i in a'_i oba hkrati ne pripadata S_1. (V nasprotnem primeru bi veljalo $f_{max}(S_1) \geq 2B + 1$.)

Definirajmo $X = S_1 \cap A$ in $Y = A/S_1$. X in Y sta razdelitev A. (Ker velja $X, Y \subseteq A$ in $X \cup Y = S_1 \cap A \cup A/S_1 = A$.) Definirajmo še $Y' = S_1 \cap A'$ in izračunajmo

$$\sum_{a_i \in X} w_i = \sum_{a_i \in S_1 \cap A} c_{12}(a_i, a_{2i}) \leq f_{12}(S_1, A_2) \leq B.$$

Izračunajmo še

$$\sum_{a_i \in Y} w_i = \sum_{a'_i \in Y'} w_i = \sum_{a'_i \in S_1 \cap A'} c_{13}(a'_i, a_{3i}) \leq f_{13}(S_1, A_3) \leq B.$$

Nadalje, ker velja

$$\sum_{a_i \in X} w_i + \sum_{a_i \in Y} w_i = \sum_{a_i \in A} w_i = 2B,$$

sledi

$$\sum_{a_i \in X} w_i = \sum_{a_i \in Y} w_i = B.$$

($\Rightarrow$) Naj bosta $X, Y \subseteq A$ rešitev problema RAZDELITEV. Definirajmo $X' = \{a'_i \in A' | a_i \in X\}$ in $Y' = \{a'_i \in A' | a_i \in Y\}$, t.j. X' in Y' vsebujeta dvojnike ustreznih elementov iz X oz. Y. Definirajmo $S_1 = X \cup Y'$ in izračunajmo $f_{max}(S_1)$. Po definiciji velja $f_{max}(S_1) = \max[f_{12}(S_1, A_2), f_{13}(S_1, A_3)]$.

Nadalje sklepajmo za $f_{12}(S_1, A_2)$,

$$\begin{aligned} f_{12}(S_1, A_2) &\leq \sum_{a_i \in X} c_{12}(a_i, a_{2i}) + \sum_{a'_i \in Y'} c_{12}(a'_i, a_{2i}) = \\ &= \sum_{a_i \in X} c_{12}(a_i, a_{2i}) = \\ &= \sum_{a_i \in X} w_i = \\ &= B. \end{aligned}$$

In podobno še za $f_{13}(S_1, A_3)$,

$$\begin{aligned} f_{13}(S_1, A_3) &\leq \sum_{a_i \in X} c_{13}(a_i, a_{3i}) + \sum_{a'_i \in Y'} c_{13}(a'_i, a_{3i}) = \\ &= \sum_{a_i \in Y'} c_{13}(a_i, a_{3i}) = \\ &= \sum_{a_i \in Y'} w_i = \\ &= \sum_{a_i \in Y} w_i = \\ &= B. \end{aligned}$$

Torej velja $f_{max}(S_1) \leq B$. □

In še končni rezultat.

Trditev 4.39 ▶ NAJMANJŠA max-PRILAGODLJIVOST p-ATRIBUTOV *na dvostopenjskem grafu prehodov scenarijev, kjer* $n = 3$, *je* $\mathcal{NP}$*-težek optimizacijski problem.*

Aproksimacijski algoritem za $n = 3$ scenarije

Oglejmo si, kako lahko enostavno iz algoritma 4.13 za NAJMANJŠE 2-STOPENJSKO $\sum$-PRILAGODLJIVO INVESTIRANJE pridobimo 2-aproksimacijski algoritem za NAJMANJŠE 2-STOPENJSKO max-PRILAGODLJIVO INVESTIRANJE, pri čemer v nalogi (G, I, C, p) nastopajo $n = 3$ scenariji. Takšen algoritem temelji na posledici 4.23.

Trditev 4.40 ▶ *Algoritem 4.13 je polinomski 2-aproksimacijski algoritem za* NAJMANJŠA max-PRILAGODLJIVOST p-ATRIBUTOV *na dvostopenjski nalogi investiranja s tremi scenariji.*

Dokaz. Število povezav v dvostopenjskem grafu prehodov scenarijev $G = (V, E)$, v katerem nastopajo $n = 3$ scenariji, je $|E| = n - 1 = 2$. Če upoštevamo posledico 4.23, neposredno dobimo željeni rezultat. □

Neaproksimabilnost za $n \geq 4$

Trditev 4.41 ▶ *Za problem* NAJMANJŠE 2-STOPENJSKO max-PRILAGODLJIVO INVESTIRANJE*, v katerem nastopajo štirje ali več ($n \geq 4$) scenarijev, ne obstaja polinomski aproksimacijski algoritem s konstantnim aproksimacijskim faktorjem, razen če $\mathcal{P}$=$\mathcal{NP}$.*

Dokaz. Neaproksimabilnost NAJMANJŠE 2-STOPENJSKO max-PRILAGODLJIVO INVESTIRANJE bomo dokazali s pomočjo konstrukcije iz dokaza trditve 4.35. (V katerem gre za dokaz $\mathcal{NP}$-polnosti NAJMANJŠE 2-STOPENJSKO $\sum$-PRILAGODLJIVO INVESTIRANJE.) V omenjenem dokazu zamenjamo c_{12} z naslednjo

$$c_{12}(v_{ijk}, a_l) = \begin{cases} 1 & i = l \\ 1 + \rho p & \text{sicer.} \end{cases}$$

Podobno naredimo še za $c_{13}(v_{ijk}, a_l)$ in $c_{14}(v_{ijk}, a_l)$.

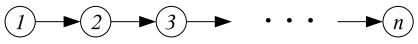
Nato iz naloge x za 3D-UJEMANJE ustvarimo nalogo x' za NAJMANJŠE 2-STOPENJSKO max-PRILAGODLJIVO INVESTIRANJE. Rešitev naloge x' ima vrednost p, če in samo če v x obstaja 3D-ujemanje. V tem primeru ima naslednja najmanjša približna rešitev vrednost $p(1 + \rho)$. V nasprotnem primeru, če v x ne obstaja 3D-ujemanje, potem ima optimalna rešitev naloge x vrednost vsaj $p(1 + \rho)$. Torej, če bi imeli polinomski ρ-aproksimacijski algoritem A za problem NAJMANJŠE 2-STOPENJSKO $\sum$-PRILAGODLJIVO INVESTIRANJE, bi z njim lahko v polinomskem času reševali problem 3D-UJEMANJE na naslednji način. Algoritem A bi pognali na nalogi x', ki ustreza nalogi x za 3D-UJEMANJE, in odgovorili z DA, če in samo če bi A vrnil rešitev vrednosti p. □

4.7 Verižno prilagajanje

4.7.1 Prevedba

V tem razdelku bomo posvetili pozornost problemom prilagajanja, v katerih je graf prehodov scenarijev *veriga*. Veriga je drevo, v katerem imajo vsa vozlišča stopnjo 2, razen vozlišč 1 in n, katerih stopnja je 1. Primer verige na n vozliščih je prikazan na sliki 4.16. Gre za preprost graf, ki je uporaben za modeliranje procesov, katerih potek je zaporeden. Izkaže se, da je problem NAJMANJŠA $\sum$-PRILAGODLJIVOST p-ATRIBUTOV na takšnem grafu optimalno rešljiv v polinomskem času. Za problem bomo izpeljali algoritem s pomočjo problema NAJMANJŠA CENA PRETOKA. Najprej bomo opisali konstrukcijo prevedbe, nato pa bomo dokazali njeno pravilnost.

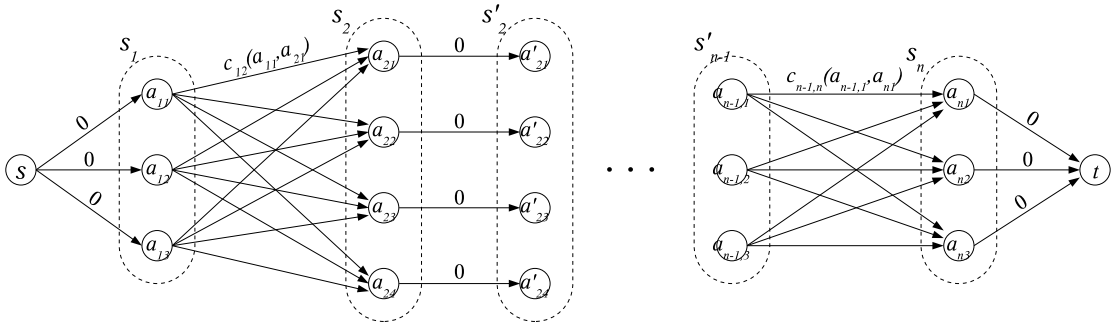
Slika 4.16: Verižni graf prehodov scenarijev.



Konstrukcija

Naj bo (G, I, C, p) naloga problema NAJMANJŠA $\sum$-PRILAGODLJIVOST p-ATRIBUTOV, kjer je $G = (V, E)$ veriga. Iz naloge (G, I, C, p) bomo skonstruirali omrežje $H = (U, F)$ za problem NAJMANJŠA CENA PRETOKA. Iz rešitve (f_{uv}) naloge H bomo nato sestavili rešitev $(S_1, S_2, \ldots, S_n)$ za nalogo (G, I, C, p). Konstrukcija je grafično prikazana na sliki 4.17.

Slika 4.17: Reševanje s pretokom.



Najprej ustvarimo dvojnike vseh scenarijev, razen s_1 in s_n, in s tem tudi dvojnike atributov. Dvojnike označimo s črtico; tako za $1 < i < n$ scenariju s_i ustreza s'_i in atributu $a_{ik} \in A_i$ ustreza $a'_{ik} \in A'_i$. Vozlišča grafa $H = (U, F)$ naj sestavljajo vsi atributi, vključno s podvojenimi, poleg tega pa še dve dodatni vozlišči s in t. Natančneje,

$$U = \{s, t\} \cup \bigcup_{1 \le i \le n} A_i \cup \bigcup_{1 < i < n} A'_i.$$

Na ta način smo ustvarili kvečjemu $2 + 2nr - 2r$ vozlišč; velja torej $|U| = O(nr)$.

Dodajmo v H še povezave. Naj velja $A'_1 = A_1$. Med vozlišči–atributi v A'_i in A_{i+1}, kjer $1 \leq i < n$, ustvarimo poln dvodelni graf, med atributi v A_i in A'_i, $1 < i < n$, pa povežimo atribut z njegovim dvojnikom. Poleg tega dodajmo še povezave iz s v atribute v A_1 in iz atributov v A_n v t. Natančneje,

$$\begin{aligned} E \;=\; & \bigcup_{1 \leq i < n} A'_i \times A_{i+1} \cup \\ & \cup\{(a_{ik}, a'_{ik}) | a_{ik} \in A_i, a'_{ik} \in A'_i, 1 < i < n\} \cup \\ & \cup\{(s, a_{1k}) | a_{1k} \in A_1\} \cup \\ & \cup\{(a_{nk}, t) | a_{nk} \in A_n\}. \end{aligned}$$

Na ta način smo ustvarili kvečjemu $(n-1)r^2 + (n-2)r + 2r$ povezav. Torej $|E| = O(nr^2)$.

Kapaciteto vseh povezav nastavimo na 1, ceno vseh povezav pa na 0, razen povezav v $A'_i \times A_{i+1}$ (polnih dvodelnih podgrafih), kjer naj bo cena povezave (a'_{ik}, a_{jl}) enaka $c_{ij}(a_{ik}, a_{jl})$.

Naj bo (f_{uv}) najcenejši pretok vrednosti p v H. Pretok (f_{uv}) v H poteka po disjunktnih poteh. Naj bodo S_i vozlišča iz A_i, kjer $1 \leq i \leq n$, skozi katera gredo te disjunktne poti. Potemtakem je $(S_1, S_2, \ldots, S_n)$ optimalna rešitev naloge (G, I, C, p). Pokažimo to bolj formalno.

Pravilnost

V splošnem ni nujno, da za problem Najmanjša Cena Pretoka obstaja vsaj ena dopustna rešitev, t.j. pretok vrednosti p. Vendar za graf $H = (U, F)$ (sestavljen z zgornjo prevedbo) pretok vrednosti p vedno obstaja. Zapišimo to v obliki leme in jo dokažimo.

Lema 4.42 ▶ *Naj bo naloga $H = (U, F)$ konstruirana iz (G, I, C, p) na zgoraj opisani način. Za nalogo $H = (U, F)$ problema* Najmanjša Cena Pretoka *vedno obstaja pretok (f_{uv}) z vrednostjo p. Poleg tega za vsak $(u, v) \in F$ velja $f_{uv} \in \mathbb{Z}^+$.*

Dokaz. V grafu H vedno obstaja množica P, kjer $|P| = p$, vozliščno disjunktnih poti iz s v t. Npr. množica

$$P = \{(s, a_{1i}, a_{2i}, a'_{2i}, \ldots, a_{n-1,i}, a'_{n-1,i}, a_{ni}, t) | 1 \leq i \leq p\}$$

je ena izmed takšnih. Iz P lahko ustvarimo rešitev (f_{uv}) naloge H na naslednji način. Za $(u, v) \in F$ naj bo $f_{uv} = 1$, če (u, v) pripada kateri izmed poti v P; sicer naj bo $f_{uv} = 0$. Pretok torej potuje samo po povezavah, ki pripadajo P. Ker so poti P vozliščno disjunktne, so tudi povezavno disjunktne. Potemtakem sledi, da je vrednost pretoka (f_{uv}) enaka p. Poleg tega so tako definirani f_{uv}, kjer $(u, v) \in F$, očitno tudi celoštevilski. □

Če obstaja dopustna rešitev, prav gotovo obstaja tudi optimalna rešitev. Zaradi trditve 2.7 o celoštevilskosti pretoka, obstaja optimalna rešitev (f_{uv}), kjer $f_{uv} \in \mathbb{Z}^+$ za vse $(u, v) \in F$. Naj bo (f_{uv}) takšna optimalna rešitev. Poleg tega definirajmo še

$$M_i = \{(u, v) \in A'_i \times A_{i+1} | f_{uv} > 0\}, \quad 1 \leq i < n$$

t.j. množica povezav med atributi v A'_i in A_{i+1}, skozi katere gre pretok $f_{uv} > 0$. (Spomnimo, da velja $A'_1 = A_1$.) Velja naslednja lema.

Lema 4.43 ▶ *Za optimalno rešitev (f_{uv}) naloge H problema* Najmanjša Cena Pretoka*, za vsak $1 \leq i < n$ velja $|M_i| = p$.*

Dokaz. Skozi ves graf gre pretok vrednosti p. Torej gre pretok vrednosti p tudi skozi vsakega izmed polnih dvodelnih podgrafov, ki ga tvorijo povezave $A'_i \times A_{i+1}$. Potemtakem za vsak $1 \leq i < n$ velja naslednje:

$$\begin{aligned} p &= \sum_{(u,v) \in A'_i \times A_{i+1}} f_{uv} = \\ &= \sum_{(u,v) \in M_i} f_{uv} = \\ &= \sum_{(u,v) \in M_i} 1 = \\ &= |M_i|. \end{aligned}$$

Pri tem smo upoštevali definicijo M_i in $f_{uv} \in \{0, 1\}$. □

Pokazali smo, da gre pretok v optimalni rešitvi v vsakem od polnih dvodelnih podgrafov, ki ga tvorijo povezave $A'_i \times A_{i+1}$, skozi natanko p povezav. Potemtakem gre torej pretok skozi natanko p vozlišč v vsaki od skupin A_i in A'_i. Velja naslednja posledica.

Posledica 4.44 ▶ *(Celoštevilčna) optimalna rešitev (f_{uv}) naloge H problema* Najmanjša Cena Pretoka *predstavlja p vozliščno-disjunktnih poti iz s v t.*

Naj bodo $S_i \subseteq A_i$, kjer $1 \leq i \leq n$, vozlišča iz skupin A_i, skozi katera gredo te poti. Prav gotovo je $(S_1, S_2, \ldots, S_n)$ dopustna rešitev naloge (G, I, C, p). Vsak pretok (f_{uv}) v H torej predstavlja neko rešitev $(S_1, S_2, \ldots, S_n)$ naloge (G, I, C, p) in obratno. Pokažimo še, da je cena pretoka (f_{uv}) enaka ceni prilagajanja $(S_1, S_2, \ldots, S_n)$. Zapišimo to v obliki naslednje leme.

Lema 4.45 ▶ *Naj bo naloga $H = (U, F)$ za problem* Najmanjša Cena Pretoka *sestavljena iz naloge (G, I, C, p) za problem* Najmanjša $\sum$-Prilagodljivost p-Atributov *in naj bo $(S_1, S_2, \ldots, S_n)$ rešitev naloge (G, I, C, p) sestavljena iz rešitve (f_{uv}) za nalogo $H = (U, F)$. Velja $f(S_1, S_2, \ldots, S_n) = cost((f_{uv}))$.*

Dokaz.

$$\begin{aligned}
cost((f_{uv})) &= \sum_{(u,v)\in F} f_{uv}w_{uv} = \\
&= \sum_{i=1}^{n-1} \sum_{(u,v)\in A'_i \times A_{i+1}} f_{uv}w_{uv} = \\
&= \sum_{i=1}^{n-1} \sum_{(u,v)\in M_i} f_{uv}w_{uv} = \\
&= \sum_{i=1}^{n-1} \sum_{(u,v)\in M_i} w_{uv} = \\
&= \sum_{i=1}^{n-1} \sum_{(u,v)\in M_i} c_{i,i+1}(u,v) = \\
&= \sum_{i=1}^{n-1} f_{i,i+1}(S_i, S_j) = \\
&= \sum_{(i,j)\in E} f_{i,j}(S_i, S_j) = \\
&= f(S_1, S_2, \ldots, S_n).
\end{aligned}$$

□

Iz pravkar dokazane leme neposredno sledi naslednji izrek.

Trditev 4.46 ▶ *Če je (f_{uv}) optimalni pretok vrednosti p v omrežju $H = (U, F)$ problema* Najmanjša Cena Pretoka, *potem je tudi iz pretoka (f_{uv}) sestavljena $(S_1, S_2, \ldots, S_n)$ optimalna rešitev za nalogo (G, I, C, p) problema* Najmanjša $\sum$-Prilagodljivost p-Atributov, *v kateri je G veriga.*

4.7.2 Algoritem

Na podlagi opisane konstrukcije lahko zapišemo algoritem za reševanje problema Najmanjša $\sum$-Prilagodljivost p-Atributov, v katerem je graf prehodov scenarijev veriga. Iz naloge (G, I, C, p) je potrebno konstruirati omrežje $H = (U, F)$, na katerem poženemo algoritem za Najmanjša Cena Pretoka, nato pa iz rešitve tega skonstruiramo rešitev $(S_1, S_2, \ldots, S_n)$ naloge (G, I, C, p). Postopek je zapisan v obliki naslednjega algoritma.

Algoritem 4.14: Verižno prilagajanje

Vhod: Naloga (G, I, C, p) prilagodljivih atributov, kjer je G veriga.
Izhod: Rešitev $(S_1, S_2, \ldots, S_n)$.
1. Iz (G, I, C, p) sestavi omrežje $H = (U, F)$;
2. V H poišči najcenejši pretok (f_{uv}) vrednosti p;
3. Iz (f_{uv}) sestavi $(S_1, S_2, \ldots, S_n)$;
4. return $(S_1, S_2, \ldots, S_n)$;

Konstrukcija naloge H ima sicer časovno zahtevnost $O(nr^2)$, vendar z izbiro primernih podatkovnih struktur za predstavitev (G, I, C, p) lahko poizvedbe v H opravljamo neposredno v (G, I, C, p). Časovna zahtevnost problema Najmanjša Cena Pretoka je $O(|U|^3)$. Velja $|U| = O(nr)$, torej drugi korak zahteva $O((nr)^3)$ časa. Konstrukcijo $(S_1, S_2, \ldots, S_n)$ lahko izvedemo v času $O(nr)$. Torej je celotna časovna zahtevnost algoritma $O(n^3r^3)$.

Na podlagi posledice 4.23 velja naslednja trditev.

Trditev 4.47 ▶ *Če algoritem 4.14 uporabimo na nalogi (G, I, C, p) za problem* Najmanjša max-Prilagodljivost p-Atributov*, v kateri je G veriga, potem za rešitev, ki jo algoritem vrne, velja $f(S_1, S_2, \ldots, S_n) \leq |E| f(S_1^*, S_2^*, \ldots, S_n^*)$.*

Algoritem 4.14 je torej tudi $|E|$-aproksimacijski algoritem za Najmanjša max-Prilagodljivost p-Atributov, v katerem je G veriga.

4.8 Aplikacije

V tem razdelku bomo predstavili nekaj morebitnih aplikacij problemov prilagodljivih atributov. Oglejmo si naslednje aplikacije:

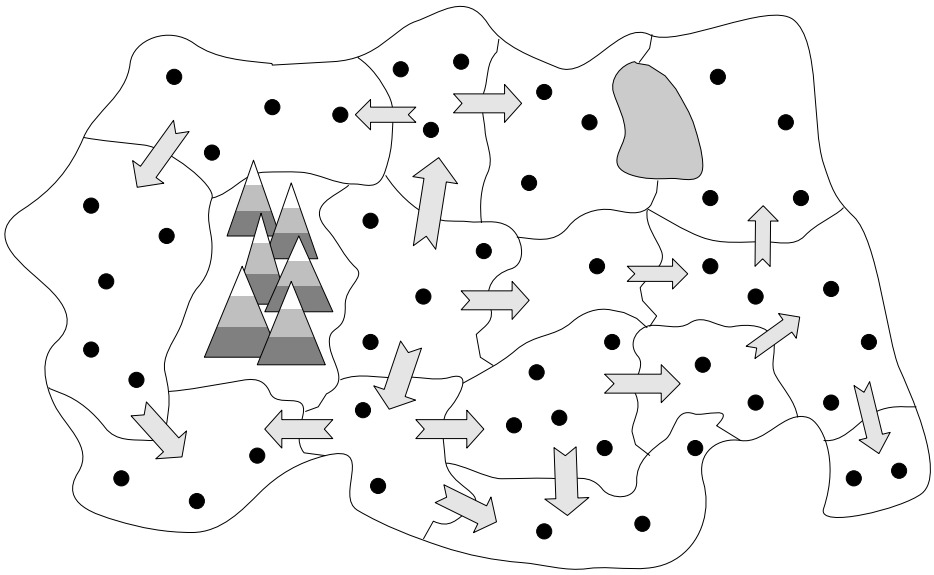
- optimizacija distribucije;
- prevajanje naravnih jezikov;
- razvoj programske opreme;
- investiranje.

4.8.1 Optimizacija distribucije

Predstavljajmo si naslednjo situacijo. Podjetje je razvilo novo vrsto izdelka, s katero želi nastopiti na trgu. V ta namen namerava v različnih regijah širom po svetu postaviti različne ponudnike, t.j. distribucijske ali trgovske centre, proizvodne obrate itd., vendar v vsaki od regij le enega. S tem skuša podjetje zagotoviti svojo prisotnost v vseh regijah, kakor tudi porazdelitev izdelave in prodaje izdelka.

V vsaki od regij ima podjetje na voljo več možnih lokacij za postavitev ponudnika. Zaradi naravnih danosti ter gospodarskih in drugih omejitev distribucija ni mogoča med poljubnim parom regij, ampak samo med določenimi regijami. Primer regij, lokacij v regijah in načinov distribuiranja je prikazan na sliki 4.18.

Slika 4.18: Distribucija izdelkov.



Poleg tega je seveda poznana tudi cena distribucije med lokacijama v dveh različnih regijah. Cilj podjetja je torej v vsaki od regij postaviti ponudnika, tako da je celotna cena distribuiranja izdelkov najmanjša.

V problemu prilagodljivih atributov regije predstavimo s scenariji, pripadajoče lokacije z atributi, cena prilagajanja dveh atributov pa je enaka ceni distribucije izdelkov med ustreznima lokacijama. Podjetje se lahko odloči za minimizacijo cene celotne distribucije, kjer gre za sum prilagodljivost, ali pa morda želi minimizirati le največjo ceno distribucije med poljubnima dvema regijama. V praksi se pogosto dogaja, da način distribuiranja ustreza drevesnemu grafu, v tem primeru pa imamo na voljo celo polinomski natančni algoritem za izračun optimalnih lokacij.

4.8.2 Prevajanje naravnih jezikov

Oglejmo si, kako lahko probleme prilagodljivih atributov uporabimo pri razreševanju dvoumnosti pri prevajanju naravnih jezikov. Dvoumnost pri prevajanju se pojavi, kadar je možnih več različnih interpretacij danega stavka. Delimo jo na leksikalno, sintaktično in semantično dvoumnost [BFJ94]. Oglejmo si primer razreševanja leksikalne[15] dvoumnosti pri prevajanju stavka “Sosednja hiša je velika.” v angleščino. Eden izmed načinov za prevajanje danega stavka je, da za vsako besedo v stavku

[15]Dvoumnost zaradi večpomenskosti besed.

poiščemo možne prevode [QGE02]. Posamezne besede in nekateri njihovi prevodi za dani primer so našteti v tabeli 4.19.

Tabela 4.19: Prevajanje naravnih jezikov.

slovenska beseda	angleški prevodi
sosednja	adjacent, adjoining, cotiguous, neighbouring, next-door
hiša	building, bungalow, cottage, dwelling, habitation, house, hut, mansion, tenement, log cabin
je	to be, to exist, to live, to take place, to eat, to take food, to munch
velika	big, large, great, tall, high, vast, spacious, capital, much, many, great deal, lot, plenty

Kot vidimo, lahko z naključno izbiro prevodov tvorimo precej nesmiselne prevode, kot je npr. "Next-door mansion eats tall.". Za posamezne prevode različnih besed lahko ugotovimo frekvenco skupne pojavitve prevodov v npr. neki knjižni zbirki leposlovnih del. Potrebno je izbrati tiste prevode, ki se največkrat pojavljajo skupaj. Na ta način upamo, da bo stavek kar najbolj smiseln.

Scenariji predstavljajo besede v stavku, atributi pa prevode posameznih besed. Cena prilagajanja dveh atributov je enaka frekvenci skupne pojavitve atributov, oz. ker gre pri problemih prilagodljivih atributov za minimizacijo, je cena pravzaprav enaka neki dovolj veliki konstanti, od katere odštejemo frekvenco.

Če imamo na voljo frekvence pojavitve za vse pare prevodov različnih besed, potem gre za polni graf prehodov scenarijev. Lahko pa scenarije povežemo v verižni graf, kot veleva zaporedje besed v stavku. Tretja možnost je, da stavek sintaktično analiziramo in v grafu prehodov scenarijev povežemo besede, med katerimi nastopajo skladenjska razmerja; tako npr. povežemo osebek s predmetom, jedro osebka s prilastkom itd. Z izbiro ustrezne knjižne zbirke za ugotavljanje frekvenc lahko izbiramo celo slog prevodov; tako lahko generiramo prevode v pravnem slogu ali v slogu Shakespeara.

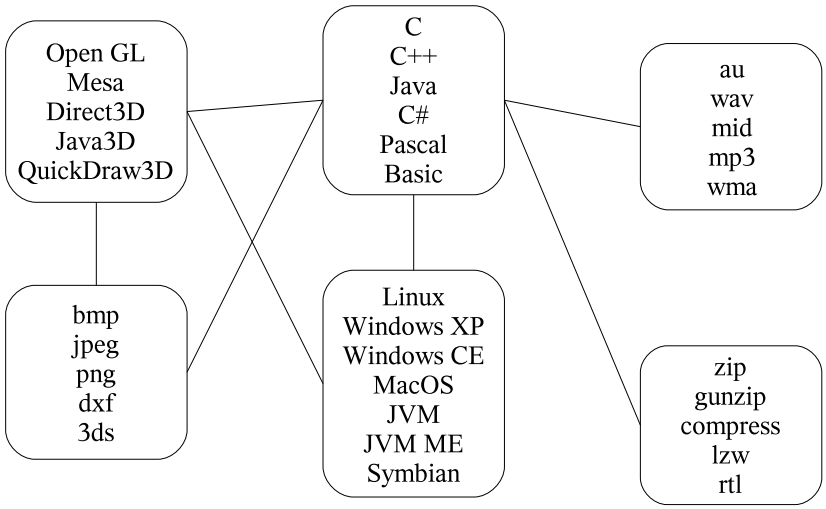
4.8.3 Razvoj programske opreme

Oglejmo si postopek razvoja programske opreme. Pri razvoju vsake programske opreme je potrebno realizirati neke funkcionalnosti. Za vsako funkcionalnost je v praksi pogosto na voljo več različnih možnosti za njeno realizacijo oz. implementacijo. Npr. kompresijo podatkov lahko izvedemo z različnimi algoritmi za kompresijo, slikovne podatke lahko zapišemo v različnih grafičnih formatih, za izris grafike lahko uporabljamo različne knjižnice itd.

Za realizacijo neke funkcionalnosti seveda ni smiselno izbrati vseh načinov, ampak le enega. Po drugi strani pa niso vsi načini realizacije neke funkcionalnosti enako dobro združljivi z načini realizacije neke druge funkcionalnosti. Zato je smiselno načine realizacije posameznih funkcionalnosti izbrati tako, da so med seboj kar

najbolj združljivi. Najbolj združljive načine lahko poiščemo s pomočjo problemov prilagodljivih atributov.

Slika 4.20: Funkcionalnosti programske opreme.



Na sliki 4.20 se nahaja primer različnih funkcionalnosti, ki jih je potrebno realizirati v neki programski opremi, kakor tudi nekatere možnosti za njihovo realizacijo. Npr. sklop ciljni operacijski sistem, programski jezik za implementacijo, knjižnica za 3D grafično okolje, grafični datotečni formati itd. Vsako od funkcionalnosti predstavimo s scenarijem, opisanim z atributi, ki predstavljajo možnosti realizacije funkcionalnosti. Tako so npr. v sklopu programskega jezika atributi: C, C++, Java, C#, Pascal, Basic itd., pri čemer je npr. programski jezik C združljiv z grafičnima knjižnicama OpenGL in Direct3D, medtem ko je njegova združljivost s Java3D in QuickDraw3D vprašljiva.

Scenarije povežemo v graf prehodov scenarijev, pri čemer je med dvema scenarijema povezava, če je med ustreznima funkcionalnostma zahtevana združljivost. Cena prilagajanja dveh atributov – funkcionalnosti lahko torej predstavlja ceno združljivosti teh dveh funkcionalnosti. V vsakem scenariju je potrebno izbrati enega izmed načinov realizacije funkcionalnosti, tako da so izbrane funkcionalnosti kar najceneje združljive.

4.8.4 Načrtovanje proizvodnje

Predstavimo še primer aplikacije, ki je podlaga za dvostopenjske probleme investiranja, ki so opisani v razdelku 4.6. Zopet si predstavljajmo podjetje, ki ima v lasti določeno število proizvodnih obratov. V vsakem od njih lahko proizvaja eno vrsto izdelka, pri čemer je možno vrsto proizvajanega izdelka spremeniti. Seveda takšna sprememba s seboj prinaša neke stroške.

V bližnji prihodnosti podjetje pričakuje spremembe na trgu, pri čemer strokovnjaki napovedujejo možno realizacijo več različnih situacij, vendar se vnaprej ne ve,

katera od njih se bo v resnici realizirala. Za vse morebitne situacije je podjetje že ugotovilo, kateri nabor izdelkov je najbolj dobičkonosen za proizvodnjo. Težava, ki se pojavi, je v tem, da podjetje želi že v sedanjosti proizvajati, vendar takšen nabor izdelkov, ki bo omogočal kar najlažji prehod v prihodnost.

Nastalo zagato lahko razrešimo s problemom prilagodljivih atributov. Sedanja in prihodnje situacije so predstavljene s scenariji, atributi pa predstavljajo izdelke. Z izborom atributov v sedanjem scenariju torej izberemo izdelke za sedanjo proizvodnjo. Cena prilagajanja atributov je enaka stroškom, ki jih prinese sprememba proizvodnje enega izdelka na drugega.

Poglavje 5

Prilagodljivostni optimizacijski problemi

Če bi lev lahko govoril, ga ne bi razumeli.
– Ludwig Wittgenstein.

Celotno poglavje o prilagodljivostnih optimizacijskih problemih je posvečeno vpeljavi prilagodljivosti v nekatere znane optimizacijske probleme. Predstavimo več prilagodljivostnih različic razmeščanja centrov in prilagodljivo najmanjše vpeto drevo. Za vse probleme opišemo zahtevnost reševanja. Poleg tega predstavimo tudi različne optimalne, hevristične in aproksimacijske algoritme za njihovo reševanje.

5.1 Uvod

V poglavju 3 smo opisali način, kako iz optimizacijskega problema P zgradimo njegovo prilagodljivostno različico P'. V tem poglavju pa bomo ta postopek prikazali na nekaterih znanih optimizacijskih problemih. Za vsakega od tako zgrajenih prilagodljivostnih problemov bomo opisali zahtevnost reševanja ter predlagali algoritme za njihovo reševanje.

Naloga (G, I, C) prilagodljivostnega problema P' je sestavljena iz nabora scenarijev $s_1, s_2, \ldots, s_n \in I$, pri čemer s_i pripada vozlišču $i \in V$, kjer je $G = (V, E)$ graf prehodov scenarijev. Znan je način izračuna partikularne kakovosti $z_i(S_i)$ partikularne rešitve S_i za scenarij s_i, t.j. $z_i(S_i) = m(s_i, S_i)$, kjer je $m(x, y)$ kriterijska funkcija problema P. Poleg tega je znan tudi način izračuna partikularne prilagodljivosti $f_{ij}(S_i, S_j)$ partikularnih rešitev S_i in S_j, kjer $(i, j) \in E$. Rešitev S_i navadno vsebuje neke elemente, kot so npr. vozlišča, povezave itd, ali pa predstavlja neko strukturo, kot npr. podgraf, drevo itd. Označimo z $|S_i|$ velikost S_i. V nadaljevanju se bomo ukvarjali s problemi, v katerih bo veljalo $|S_i| = |S_j|$ za vse $(i, j) \in E$. Množica C pa vsebuje cene $c_{ij}(u, v)$, kjer $(i, j) \in E$, prilagajanja dveh elementov

$u \in S_i$ in $v \in S_j$. Na ta način lahko f_{ij} izračunamo enako kot v problemih prilagodljivih atributov.

V P' vedno nastopata oba kriterija vrednotenja rešitev, tako partikularna kakovost kot partikularna prilagodljivost. Pri tem partikularna (oz. celotna) prilagodljivost nastopa kot del kriterijske funkcije problema P', medtem ko partikularna kakovost lahko v P' nastopa na različne načine, kot so npr.:

- omejitve $z_i(S_i) \leq \alpha_i z_i^*$, kjer $i \in V$ (v primeru minimizacijskega problema P);
- omejitev $z(S_1, S_2, \ldots, S_n) \leq \alpha z^*$;
- del kriterijske funkcije (večkriterijska optimizacija).

V nadaljevanju bomo izpeljali prilagodljivostne različice naslednjih optimizacijskih problemov:

- *razmeščanje centra v omrežju* – zanj obstaja polinomsko število optimalnih rešitev, kar omogoča, da lahko prilagodljivostno različico problema neposredno rešujemo s pomočjo problemov prilagodljivih atributov;
- *najmanjše vpeto drevo* – v polinomskem času optimalno rešljiv problem, za katerega obstajajo požrešni algoritmi, iz katerih lahko skonstruiramo hevristične algoritme za reševanje prilagodljivostne različice;
- *razmeščanje centrov v omrežju* – gre za $\mathcal{NP}$-težek problem, zanj bomo izpeljali dve prilagodljivostni različici, pri tem bo v eni kakovost nastopala kot omejitev in v drugi kot del kriterijske funkcije.

Prilagodljivostne različice vseh naštetih problemov so $\mathcal{NP}$-težki problemi, kar bomo v nadaljevanju tudi pokazali.

Odnos med P in P'

Če je P $\mathcal{NP}$-težek problem, to še ne pomeni, da je njegova prilagodljivostna različica P' tudi $\mathcal{NP}$-težka. Razlog za to je, da $z_i(S_i)$ v P' lahko nastopa kot omejitev, t.j. $z_i(S_i) \leq \alpha_i z_i^*$, kar v primeru $P \in APX$ pomeni, da z izbiro dovolj visoke α_i tej omejitvi lahko zadostimo.

Pravi razlog $\mathcal{NP}$-težkosti P' se navadno skriva v zahtevnosti iskanja najceneje prilagodljivih partikularnih rešitev, gre torej za enak razlog kot pri problemih prilagodljivih atributov. Pri dokazovanju $\mathcal{NP}$-težkosti se torej največkrat opiramo na znane rezultate o $\mathcal{NP}$-težkosti problemov prilagodljivih atributov iz predhodnega poglavja.

Osnovni problem, ki se skriva v ozadju problemov prilagodljivih atributov, je zelo preprost – gre le za izbiro poljubnega atributa. Zatorej je verjetno, da je vsak prilagodljivostni problem P', ki je izpeljan iz nekega optimizacijskega problema P, $\mathcal{NP}$-težek na splošnem grafu prehodov scenarijev. Za dokaz $\mathcal{NP}$-težkosti P je potrebno le poiskati prevedbo iz nekega problema prilagodljivih atributov.

5.2 Prilagodljivi 1-center

5.2.1 Vpeljava prilagodljivosti

Definicija osnovnega problema

V tem razdelku se bomo najprej lotili obravnave problema Najmanjši 1-Center. Izpeljali bomo prilagodljivostno različico problema in pokazali njeno $\mathcal{NP}$-težkost. Pokazali bomo, kako lahko za reševanje problema uporabimo rezultate iz poglavja o problemih prilagodljivih atributov.

Problem Najmanjši 1-Center izhaja s področja razmeščanja ponudnikov. Iz literature je znanih več različic problema [Mih01, Mih04a], na tem mestu pa se bomo osredotočili na različico, ki je definirana na omrežju. Vhodni podatek problema je poln neusmerjen graf $G = (V, E)$, kjer je za vsako povezavo $(u, v) \in E$ podana njena dolžina $d(u, v)$. Problem Najmanjši 1-Center je poiskati vozlišče $c \in V$, tako da je največja razdalja od poljubnega vozlišča $v \in V$ do c najmanjša. Vozlišče c imenujemo tudi *center* grafa. Definicija problema je naslednja.

Problem 5.22: Najmanjši 1-Center

Naloga: Neusmerjeno polno omrežje $G = (V, E)$, kjer je $d(u, v)$ dolžina povezave $(u, v) \in E$.
Dopustna rešitev: $c \in V$.
Kakovost dopustne rešitve: $z(c) = \max_{v \in V} d(c, v)$.
Cilj: Minimizacija.

Moč množice dopustnih rešitev je $|V|$. Očitno je problem mogoče optimalno rešiti v polinomskem času. To dejstvo bomo izkoristili za reševanje prilagodljivostne različice problema.

Definicija prilagodljivostnega problema

Prilagodljivostno različico problema bomo vpeljali na naslednji način. Naj bo $G = (V, E)$ graf prehodov scenarijev in I množica scenarijev, kjer je scenarij $s_i \in I$ opisan s polnim omrežjem $G_i = (V_i, E_i)$. Omrežje G_i predstavlja nalogo za Najmanjši 1-Center. Prilagodljivostno različico bomo poimenovali Najmanjši $\sum$-Prilagodljivi 1-Center.

Dopustne rešitve problema Najmanjši $\sum$-Prilagodljivi 1-Center so zaporedja $(c_1, c_2, \ldots, c_n)$, kjer je $c_i \in V_i$ optimalni center grafa G_i glede na

$$z_i(c) = \max_{v \in V_i} d_i(c, v),$$

kjer je $d_i(u, v)$ dolžina povezave $(u, v) \in E_i$. Za vsak $1 \leq i \leq n$ velja torej $z_i(c_i) = z_i^*$, pri čemer je z_i^* vrednost optimalne rešitve problema Najmanjši 1-Center na omrežju G_i.

Poleg tega je za vse $(i, j) \in E$ poznana cena $c_{ij}(u, v)$ prilagajanja $u \in V_i$ in $v \in V_j$. V realnosti lahko $c_{ij}(u, v)$ predstavlja npr. ceno premestitve ponudnika z lokacije u na v pri prehodu iz scenarija i v j.

Najmanjši $\sum$-Prilagodljivi 1-Center je poiskati $(c_1, c_2, \ldots, c_n)$, tako da minimiziramo kriterij celotne prilagodljivosti

$$f_{sum}(c_1, c_2, \ldots, c_n) = \sum_{(i,j)\in E} c_{ij}(c_i, c_j).$$

Zapišimo njegovo definicijo.

Problem 5.23: Najmanjši $\sum$-Prilagodljivi 1-Center

Naloga: Trojica (G, I, C), kjer je scenarij $s_i \in I$ opisan s povezanim omrežjem $G_i = (V_i, E_i)$.
Dopustna rešitev: Zaporedje $(c_1, c_2, \ldots, c_n)$, kjer $c_i \in V_i$ in velja $z_i(c_i) = z_i^*$.
Kakovost dopustne rešitve: $f_{sum}(c_1, c_2, \ldots, c_n)$.
Cilj: Minimizacija.

V primeru max prilagodljivosti pa bo šlo za minimizacijo

$$f_{max}(c_1, c_2, \ldots, c_n) = \max_{(i,j)\in E} c_{ij}(c_i, c_j).$$

Zapišimo še njegovo definicijo.

Problem 5.24: Najmanjši max-Prilagodljivi 1-Center

Naloga: Trojica (G, I, C), kjer je scenarij $s_i \in I$ opisan s povezanim omrežjem $G_i = (V_i, E_i)$.
Dopustna rešitev: Zaporedje $(c_1, c_2, \ldots, c_n)$, kjer $c_i \in V_i$ in velja $z_i(c_i) = z_i^*$.
Kakovost dopustne rešitve: $f_{max}(c_1, c_2, \ldots, c_n)$.
Cilj: Minimizacija.

V obeh problemih gre torej za iskanje množice optimalnih centrov za vsa podana omrežja, pri čemer so centri med seboj kar najbolj prilagodljivi. Pri tem sta si lahko dve omrežji G_i in G_j, kjer $i, j \in V$, med seboj lahko poljubno različni, t.j. dovoljene so kvalitativne in kvantitativne perturbacije scenarijev. V nadaljevanju se bomo osredotočili le na najmanjši Najmanjši $\sum$-Prilagodljivi 1-Center, vendar bo vse povedano v analognem smislu veljalo tudi za Najmanjši max-Prilagodljivi 1-Center.

5.2.2 Zahtevnost reševanja

Kot smo že navajeni iz predhodnih razdelkov, pri vpeljavi novih problemov največkrat naletimo na $\mathcal{NP}$-težek problem, in tudi Najmanjši $\sum$-Prilagodljivi 1-

Center ni izjema. Zapišimo naslednjo trditev.

Trditev 5.48 ▶ Najmanjši $\sum$-Prilagodljivi 1-Center *je $\mathcal{NP}$-težek optimizacijski problem.*

Dokaz. Označimo s P problem $\sum$-Prilagodljivost 1-Atributov in s P' odločitveno različico problema Najmanjši $\sum$-Prilagodljivi 1-Center. Podali bomo oris prevedbe P na P', t.j. iz naloge (G, I, C) za P bomo ustvarili nalogo (G, I', C) za P'.

Naj bo $A_i = \{a_{i1}, a_{i2}, \ldots, a_{i|A_i|}\}$ množica atributov, ki opisuje scenarij $s_i \in I$. Ustvarimo poln graf $G_i = (A_i, E_i)$ za scenarij $s'_i \in I'$. Dolžina vseh povezav $e \in E_i$ naj bo enaka 1, t.j. za vse $(u, v) \in E_i$ velja $d(u, v) = 1$. V G_i je vsak $v \in V_i$ optimalen center. Očitno je, da sta si rešitvi naloge (G, I, C) problema P in naloge (G, I', C) problema P' enaki. □

Zapišimo še analogno trditev za max prilagodljivost.

Trditev 5.49 ▶ Najmanjši max-Prilagodljivi 1-Center *je $\mathcal{NP}$-težek optimizacijski problem.*

5.2.3 Reševanje

Najmanjši $\sum$-Prilagodljivi 1-Center lahko preprosto rešimo s pomočjo problema Najmanjša $\sum$-Prilagodljivost 1-Atributov. Oglejmo si, kako. Povedali smo, da je problem Najmanjši 1-Center optimalno rešljiv v polinomskem času; še več, v polinomskem času lahko poiščemo vse optimalne centre omrežja $G_i = (V_i, E_i)$. To naredimo preprosto tako, da za vsak $v \in V_i$ izračunamo $z_i(v)$ in izberemo vse v z najmanjšo vrednostjo. Označimo z A_i množico optimalnih centrov za G_i, t.j.

$$A_i = \{u \in V_i | z_i(u) = z_i^*\}.$$

Časovna zahtevnost izračuna A_i je $O(|V_i|^2)$.

Za problem Najmanjši $\sum$-Prilagodljivi 1-Center je torej podana naloga (G, I, C), iz katere sestavimo nalogo (G, I', C) za problem Najmanjša $\sum$-Prilagodljivost 1-Atributov. Za vsak G_i izračunamo množico A_i optimalnih centrov, ki opisuje scenarij $s'_i \in I'$. Za reševanje (G, I', C) lahko uporabimo kakega od algoritmov, opisanih v predhodnem poglavju. Rešitev naloge (G, I', C) je hkrati tudi rešitev (G, I, C). Postopek reševanja je zapisan kot naslednji algoritem.

Algoritem 5.15: Prilagodljivi 1-center

Vhod: Naloga (G, I, C).
Izhod: Zaporedje $(c_1, c_2, \ldots, c_n)$.
1. forall $s_i \in I$ do
2. $\quad A_i := \{u \in V_i | z_i(u) = \min_{v \in V_i} z_i(v)\}$;
3. Reši (G, I', C);

Časovna zahtevnost prvih dveh vrstic je $O(nr^2)$. Težavo predstavlja vrstica 3, v kateri gre še vedno za reševanje $\mathcal{NP}$-težkega problema. V primeru, da je G drevo, pa je seveda problem rešljiv v polinomskem času.

Vse povedano velja tudi za Najmanjši max-Prilagodljivi 1-Center, le da v tem primeru namesto sum prilagodljivosti uporabimo max prilagodljivost. Kot vidimo lahko s pomočjo problemov prilagodljivih atributov enostavno rešujemo problem razmeščanja centra. Pravzaprav na enak način lahko rešujemo še več optimizacijskih problemov. Tak problem je npr. tudi Najmanjša 1-Mediana.

5.3 Prilagodljivo najmanjše vpeto drevo

5.3.1 Vpeljava prilagodljivosti

Lotimo se znanega problema iskanja najmanjšega vpetega drevesa v omrežju. Gre za zanimiv problem, ki je pogosto uporaben pri snovanju omrežij. Naj bo $G = (V, E)$ povezan graf. *Vpeto drevo* $T \subseteq G$ je povezano drevo $T = (V, F)$, kjer $F \subseteq E$. Vpeto drevo torej razpenja svoje povezave preko vseh vozlišč grafa G. Definicija problema je naslednja.

Problem 5.25: Najmanjše Vpeto Drevo

Naloga: Povezano omrežje $G = (V, E)$, kjer je $w(e)$ cena povezave $e \in E$.
Dopustna rešitev: Vpeto drevo $T = (V, F)$ v G.
Kakovost dopustne rešitve: $\sum_{e \in F} w(e)$.
Cilj: Minimizacija.

Za problem obstaja več polinomskih algoritmov za natančno reševanje. Med njimi sta najbolj znana Kruskalov in Primov algoritem, ki ju je možno implementirati tako, da je njuna časovna zahtevnost $O(|E| \log |V|)$ [CLRS01, KT06]. Definirajmo še njegovo prilagodljivostno različico.

Problem 5.26: NAJMANJŠE $\sum$-PRILAGODLJIVO VPETO DREVO

Naloga: Trojica (G, I, C), kjer je scenarij $s_i \in I$ opisan s povezanim omrežjem $G_i = (V_i, E_i)$. Za vsak $i, j \in V$ velja $|V_i| = |V_j|$.
Dopustna rešitev: Zaporedje $(T_1, T_2, \ldots, T_n)$, kjer je $T_i \subseteq G_i$ najmanjše vpeto drevo.
Kakovost dopustne rešitve: $f_{sum}(T_1, T_2, \ldots, T_n)$.
Cilj: Minimizacija.

V problemu za velikost omrežij G_i, kjer $i, j \in V$, velja omejitev $|V_i| = |V_j|$. Med scenariji so torej dovoljene poljubne kvantitativne perturbacije. Pri uporabi kvalitativnih perturbacij pa moramo paziti, da ne spreminjajo števila vozlišč v omrežju.

Naj velja $T_i = (V_i, F_i)$ in $T_j = (V_j, F_j)$. Posledica omejitve je, da za rešitev $(T_1, T_2, \ldots, T_n)$ velja $|F_i| = |F_j| = |V_i| - 1$, kjer $i, j \in V$. S tem dosežemo, da je partikularna prilagodljivost $f_{ij}(T_i, T_j)$ nedvoumno definirana. Ceno prilagajanja T_i in T_j izračunamo s pomočjo problema NAJMANJŠA DODELITEV na nalogi $H = (F_i \cup F_j, F_i \times F_j)$, pri čemer povezave iz F_i in F_j v grafu H predstavljajo vozlišča. Uteži povezav (e_{ik}, e_{jl}) v H so enake cenam $c_{ij}(e_{ik}, e_{jl})$ prilagajanja povezav $e_{ik} \in E_i$ in $e_{jl} \in E_j$.

5.3.2 Zahtevnost reševanja

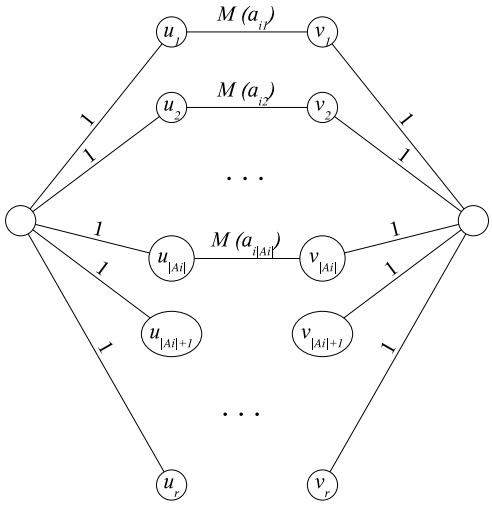
Trditev 5.50 ▶ NAJMANJŠE $\sum$-PRILAGODLJIVO VPETO DREVO *je $\mathcal{NP}$-težek optimizacijski problem.*

Dokaz. Opisali bomo prevedbo problema $\sum$-PRILAGODLJIVOST 1-ATRIBUTOV (naloga (G, I, C)) na odločitveno različico problema NAJMANJŠE $\sum$-PRILAGODLJIVO VPETO DREVO (naloga (G, I', C')). Za vsak A_i, ki opisuje scenarij $s_i \in I$, ustvarimo omrežje G_i, ki opisuje scenarij $s'_i \in I'$, kot je prikazano na sliki 5.1.

Torej $G_i = (V_i, E_i)$, kjer $A_i \subset E_i$. Vsi G_i, kjer $i \in V$, vsebujejo enako število $2(r+1)$ vozlišč, posledično so torej tudi vsa vpeta drevesa enake velikosti. V vsakem od G_i obstaja natanko $|A_i|$ različnih najmanjših vpetih dreves, pri čemer vsako od teh dreves vsebuje vse povezave s ceno 1 in eno izmed povezav s ceno $M \geq 2$, ki so v G_i označene z $a_{ik} \in A_i$. (Na sliki 5.1 se oznaka atributa nahaja na povezavi v oklepaju.)

Izbrana povezava izmed povezav v A_i predstavlja izbiro atributa iz A_i. Potrebno je le definirati $c'_{ij}(e, f)$, tako da bo $f_{ij}(T_i, T_j)$ enaka ceni pretvorbe označenih povezav (in s tem ustreznih atributov) v T_i in T_j. To naredimo tako, da definiramo $c'_{ij}(e, f)$,

Slika 5.1: Graf G_i.



kjer $e \in E_i$ in $f \in E_j$, kot

$$c'_{ij}(e,f) = \begin{cases} c_{ij}(a_{ik}, a_{jl}) & e = a_{ik} \in A_i, f = a_{jl} \in A_j \\ \infty & e \in A_i, f \notin A_j \\ \infty & e \notin A_i, f \in A_j \\ 0 & sicer. \end{cases}$$

Cena $f_{ij}(T_i, T_j)$ je s tem enaka $c_{ij}(a_i, a_j)$, kjer sta $a_i \in A_i$ in $a_j \in A_j$ povezavi, izbrani v T_i in T_j.[1] Očitno za rešitev $(a_1, a_2, \ldots, a_n)$ naloge (G, I, C) in ustrezno rešitev $(T_1, T_2, \ldots, T_n)$ naloge (G, I', C') velja $f_{sum}(a_1, a_2, \ldots, a_n) = f_{sum}(T_1, T_2, \ldots, T_n)$. □

Seveda velja podobna trditev tudi za različico max prilagodljivosti iskanja najmanjšega vpetega drevesa.

Trditev 5.51 ▶ Najmanjše max-Prilagodljivo Vpeto Drevo *je $\mathcal{NP}$-težek optimizacijski problem.*

5.3.3 Prilagodljivostni Primov algoritem

Za Najmanjše $\sum$-Prilagodljivo Vpeto Drevo bomo izpeljali hevristični algoritem iz Primovega algoritma za Najmanjše Vpeto Drevo, zato si najprej oglejmo slednjega. Algoritem gradi vpeto drevo postopoma, tako da na vsakem koraku obstoječemu drevesu doda eno povezavo. Gradnja drevesa se začne z drevesom, ki je sestavljeno iz enega samega vozlišča, nadaljuje pa tako, da se na vsakem koraku

[1] Naj bo $T_i = (V_i, F_i)$ najmanjše vpeto drevo v G_i, potem velja $\{a_i\} = A_i \cap F_i$.

doda najcenejša povezava, katere začetek je v poljubnem vozlišču drevesa, konec pa v poljubnem vozlišču, ki ne pripada drevesu. Označimo z $V(T)$ vozlišča drevesa T in zapišimo naslednji algoritem.

Algoritem 5.16: Primov algoritem

Vhod: Omrežje $G = (V, E)$.
Izhod: Najmanjše vpeto drevo $T \subseteq G$.

1. $T := (\{v\}, \emptyset)$, kjer je $v \in V$ poljuben;
2. while $V(T) \neq V$ do
3. $\quad e := \arg\min_{f=(u,v), u \in V/V(T), v \in V(T)} w(f)$;
4. $\quad T := T + e$;
5. end;
6. return T;

Dodajanje povezav v T se izvaja, dokler T ne vsebuje vseh vozlišč V (vrstica 2). V vrstici 3 se ugotovi najcenejša povezava $e \in E$, ki jo lahko dodamo T (vrstica 4).

Na vsakem koraku (v vrstici 3) je v splošnem možnih več povezav e, ki jih lahko dodamo T. To lahko izkoristimo pri gradnji algoritma za prilagodljivostno različico problema. Za vsak scenarij ugotovimo množico takšnih najcenejših povezav, iz katerih konstruiramo nalogo za problem prilagodljivih atributov. To nalogo lahko rešimo s katerim od algoritmov iz predhodnega poglavja, povezave iz rešitve pa nato dodamo v ustrezna drevesa. Zapišimo to kot naslednji hevristični algoritem.

Algoritem 5.17: Prilagodljivostni Primov algoritem

Vhod: Naloga (G, I, C).
Izhod: Zaporedje $(T_1, T_2, \ldots, T_n)$, kjer je $T_i \subseteq G_i$ najmanjše vpeto drevo.

1. forall $i \in V$ do
2. $\quad T_i := (\{v_i\}, \emptyset)$, kjer je $v_i \in V_i$ poljuben;
3. while $V(T_1) \neq V_1$ do
4. $\quad$ forall $i \in V$ do
5. $\quad\quad F_i := \{(u, v) \in E_i | u \in V(T_i), v \in V_i/V(T_i)\}$;
6. $\quad\quad A_i := \{e \in F_i | w(e) = \min_{f \in F_i} w(f)\}$;
7. $\quad$ end;
8. $\quad$ Sestavi (G, I', C), kjer je $s_i \in I'$ opisan z A_i;
9. $\quad (a_1, a_2, \ldots, a_n) :=$ reši (G, I', C);
10. $\quad$ forall $i \in V$ do
11. $\quad\quad T_i := T_i + a_i$;
12. end;
13. return $(T_1, T_2, \ldots, T_n)$;

V vrsticah 1 in 2 inicializiramo začetna drevesa za vse scenarije. Nato $(|V_i| - 1)$-krat ponovimo vrstice od 4 do 11. Na ta način v vsa T_i dodamo $|V_i| - 1$ povezav.

(Velja $|V_i| = |V_j|$ za vse $i, j \in V$.) S tem T_i postane vpeto drevo. V vrsticah od 4 do 7 izračunamo množico A_i povezav, ki jih lahko dodamo v T_i. Nato (vrstici 8 in 9) sestavimo nalogo za problem prilagodljivih atributov in jo rešimo. Rešitev $(a_1, a_2, \ldots, a_n)$ uporabimo za širitev dreves T_i, kjer $i \in V$.

Oglejmo si še časovno zahtevnost. Zanka v vrstici 3 se izvede $(|V_i| - 1)$-krat. Za gradnjo množic A_i porabimo $O(n|V_i|^2)$ časa. Če je graf G drevo, je časovna zahtevnost reševanja (G, I', C) reda $O(n^2r^2)$, kjer je $r = \max_i |A_i| = \max_i |E_i|$. Celotna časovna zahtevnost je torej $O(n^2r^2|V_i|)$. V primeru, da je G splošen graf, pa je seveda algoritem superpolinomski.

5.4 Prilagodljivo razmeščanje centrov

5.4.1 Vpeljava prilagodljivosti

V razdelku 5.2 smo obdelali Najmanjši $\sum$-Prilagodljivi 1-Center, kjer gre za razmeščanje enega centra s sposobnostjo prilagajanja. V tem razdelku si bomo ogledali posplošeno različico tega problema, v kateri gre za razmeščanje p centrov. Najprej definirajmo osnovno, t.j. neprilagodljivostno različico problema.

Problem 5.27: Najmanjši p-center

Naloga: Neusmerjeno polno omrežje $G = (V, E)$ in pozitivno celo število $p \leq |V|$.
Dopustna rešitev: Množica $C \subseteq V$, tako da $|C| \leq p$.
Kakovost dopustne rešitve: $z(S) = \max_{v \in V} \min_{c \in C} d(v, c)$.
Cilj: Minimizacija.

Gre torej za iskanje takšne podmnožice vozlišč grafa, da je razdalja poljubnega vozlišča do njegovega najbližjega centra kar najmanjša. Najmanjši p-center je $\mathcal{NP}$-težek optimizacijski problem [ACG+99]. Poleg tega spada med probleme, za katere ne obstaja aproksimacijski algoritem s konstantnim aproksimacijskim faktorjem.

Večkrat se omejimo na takšno različico problema, v kateri za povezave grafa velja trikotniška neenakost. V tem primeru problem pripada razredu $\mathcal{APX}$, zanj je znana celo spodnja meja aproksimabilnosti, t.j. za problem ne obstaja $(2 - \epsilon)$-aproksimacijski algoritem, razen če $\mathcal{P} = \mathcal{NP}$. Poleg tega je znanih več algoritmov, ki to spodnjo mejo tesno dosegajo. Več o problemu in algoritmih za njegovo reševanje je v [Mih01, Mih04a].

Lotimo se konstrukcije prilagodljivostne različice. Vsak od scenarijev $s_i \in I$ prilagodljivostne različice je opisan z omrežjem $G_i = (V_i, E_i)$. Za vse G_i, kjer $i \in V$, bomo v nadaljevanju predpostavljali veljavnost trikotniške neenakosti. Za vsakega od scenarijev s_i želimo poiskati dopustno rešitev $C_i \subseteq V_i$, kjer $|C_i| \leq p$, katere

performančni kvocient je kvečjemu 2, t.j. $R(G_i, C_i) \leq 2$; več v polinomskem času niti ne moremo pričakovati. Zapišimo definicijo sum prilagodljivostne različice.

Problem 5.28: NAJMANJŠI $\sum$-PRILAGODLJIVI p-CENTER

Naloga: Trojica (G, I, C), kjer je scenarij $s_i \in I$ opisan z omrežjem $G_i = (V_i, E_i)$, in parameter $p \leq \min_{1 \leq i \leq n} |V_i|$.
Dopustna rešitev: Zaporedje $(C_1, C_2, \ldots, C_n)$, kjer je $C_i \subseteq V_i$ in $|C_i| \leq p$. Velja $z_i(C_i) \leq 2z_i^*$.
Kakovost dopustne rešitve: $f_{sum}(C_1, C_2, \ldots, C_n)$.
Cilj: Minimizacija.

Pri tem je z_i^* vrednost optimalne rešitve za nalogo s_i problema NAJMANJŠI p-CENTER. Množica C vsebuje funkcije $c_{ij}(v_{ik}, v_{jl})$ za vse $v_{ik} \in V_i$ in $v_{jl} \in V_j$, kjer $(i, j) \in G$, s pomočjo katerih izračunamo partikularno prilagodljivost $f_{ij}(C_i, C_j)$. To naredimo enako kot pri problemih prilagodljivih atributov, le da imamo v tem primeru namesto atributov vozlišča. Problema NAJMANJŠI max-PRILAGODLJIVI p-CENTER je definiran enako, le da v njem nastopa $f_{max}(C_1, C_2, \ldots, C_n)$ kot kriterijska funkcija.

5.4.2 Zahtevnost reševanja

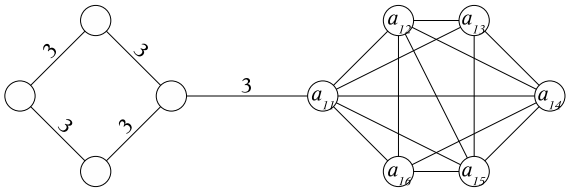
Dejstvo, da NAJMANJŠI p-CENTER spada med $\mathcal{NP}$-težke optimizacijske probleme, še ne pomeni, da enako velja tudi za NAJMANJŠI $\sum$-PRILAGODLJIVI p-CENTER. Oglejmo si prilagodljivostni problem na enem scenariju. Potrebno je torej poiskati C_1, tako da velja $z_1(C_1) \leq 2z_1^*$. To pa lahko preprosto storimo v polinomskem času z Gonzalezovim algoritmom, ki ga bomo opisali v nadaljevanju. Kljub temu za problem definiran na splošnem grafu prehodov scenarijev velja naslednja trditev.

Trditev 5.52 ▶ NAJMANJŠI $\sum$-PRILAGODLJIVI p-CENTER *je $\mathcal{NP}$-težek optimizacijski problem.*

Dokaz. V prevedbi bomo zopet uporabili $\sum$-PRILAGODLJIVOST 1-ATRIBUTOV (naloga (G, I, C)), ki ga bomo prevedli na odločitveno različico problema NAJMANJŠI $\sum$-PRILAGODLJIVI p-CENTER (naloga (G, I', C')). Za vsak A_i, ki opisuje scenarij $s_i \in I$, ustvarimo omrežje $G_i = (V_i, E_i)$, ki opisuje scenarij $s_i' \in I'$. Omrežje G_i je sestavljeno iz cikla na $p - 1$ vozliščih, ki je z eno povezavo povezano s polnim grafom na $|A_i|$ vozliščih. Vozlišča v polnem grafu predstavljajo atribute iz A_i. Cene povezav v polnem podgrafu so enake 1, cene vseh ostalih povezav pa 3. Primer G_i za $p = 5$ in $|A_i| = 6$ je prikazan na sliki 5.2.[2]

[2]NAJMANJŠI 1-CENTER je definiran na polnem omrežju, vendar lahko iz G_i preprosto izračunamo polno omrežje s pomočjo algoritma 2.1.

Slika 5.2: Graf G_i.



Optimalna in s tem tudi 2-aproksimacijska rešitev je sestavljena iz $p-1$ vozlišč, ki pripadajo ciklu in enega poljubnega vozlišča iz A_i. Množica C' je sestavljena iz funkcij $c'_{ij}(u,v)$, kjer $u \in V_i$ in $v \in V_j$, ki so definirane kot

$$c'_{ij}(u,v) = \begin{cases} c_{ij}(a_{ik}, a_{jl}) & u = a_{ik} \in A_i, v = a_{jl} \in A_j \\ \infty & u \in A_i, v \notin A_j \\ \infty & u \notin A_i, v \in A_j \\ 0 & \textit{sicer.} \end{cases}$$

Partikularna prilagodljivost $f_{ij}(C_i, C_j)$ je torej enaka $c'_{ij}(a_i, a_j)$, kjer $\{a_i\} = A_i \cap C_i$ in $\{a_j\} = A_j \cap C_j$. Očitno velja $f_{sum}(a_1, a_2, \ldots, a_n) = f_{sum}(C_1, C_2, \ldots, C_n)$. □

Zapišimo podobno trditev še za max prilagodljivost.

Trditev 5.53 ▶ Najmanjši max-Prilagodljivi p-Center *je $\mathcal{NP}$-težek optimizacijski problem.*

5.4.3 Prilagodljivostni Gonzalezov algoritem

Najprej si oglejmo Gonzalezov algoritem za reševanje problema Najmanjši p-center [Gon85]. Algoritem vozlišča za postavitev centra izbira iterativno, tako da najprej izbere naključno vozlišče, nato pa na vsakem koraku izbere vozlišče, ki je od trenutne množice centrov najbolj oddaljeno. Postopek je zapisan kot naslednji algoritem.

Algoritem 5.18: Gonzalezov algoritem

Vhod: Omrežje $G = (V, E)$ in parameter $p \in \mathbb{Z}^+$.
Izhod: Množica centrov $C \subseteq V$.

1. $C := \{c\}$, kjer je $c \in V$ poljuben;
2. while $|C| < p$ do
3. $\quad u := \arg\max_{v \in V/C} \min_{c \in C} d(c, v)$;
4. $\quad C := C \cup \{u\}$;
5. end;
6. return C;

Na začetku algoritma (vrstica 1) se izbere poljubno vozlišče in doda v C. Nato se izvaja zanka (vrstice 2 do 5), v kateri se na vsakem koraku izbere $u \in V/C$ in doda v C. Časovna zahtevnost algoritma je $O(np^2)$. Velja naslednja trditev.

Trditev 5.54 ▶ *Gonzalezov algoritem je 2-aproksimacijski algoritem za* Najmanjši p-center*, v katerem velja trikotniška neenakost.*

Dokaz. Algoritem zgradi končno rešitev $C = C_{p-1}$ v p korakih, kjer na vsakem koraku iz predhodne rešitve C_{i-1} izračuna novo trenutno rešitev C_i. Naj začetna rešitev C_0 vsebuje poljubno vozlišče $v_0 \in V$, torej $C_0 = \{v_0\}$. Naredimo še en dodatni korak algoritma, v katerem bi algoritem izbral vozlišče v_p. Njena razdalja[3] do C_{p-1} je ravno pokrivna razdalja $z(C_p) = d(v_p, C_{p-1})$ rešitve C_p. Razdalja od kateregakoli vozlišča $v \in V$ do množice C_i med algoritmom ne narašča, torej $d(v, C_i) \geq d(v, C_{i-1})$. Zaradi načina izbire v_i velja tudi $d(v_i, C_{i-1}) \geq d(v, C_{i-1})$. V množici $C_p = \{v_0, \ldots, v_p\}$ je $p + 1$ vozlišč, torej sta vsaj dve vozlišči, v_i in v_j, dodeljeni istemu centru c iz optimalne rešitve. Naj bo pokrivna razdalja optimalne rešitve z^* in naj bo $i < j$. Torej lahko zapišemo

$$z(C_p) = d(v_p, C_{p-1}) \leq d(v_j, C_{j-1}) \leq d(v_i, v_j) \leq d(c, v_i) + d(c, v_j) \leq 2z^*,$$

ker je v_i že v C_{j-1} v koraku j in zaradi veljavnosti trikotniške neenakosti. □

Iz Gonzalezovega algoritma in iz algoritma za problem prilagodljivh atributov bomo zgradili hevristični algoritem za obe prilagodljivostni različici problema. Pri tem bomo izkoristili dejstvo, da v Gonzalezovem algoritmu v vrstici 3 pravzaprav več kandidatov $u \in V/C$ za postavitev centra, t.j. več vozlišč, ki so najbolj oddaljena od trenutne množice centrov. Namesto enega bomo poiskali vse takšne kandidate. To bomo storili za vse G_i, kjer $i \in V$. Množice potencialnih kandidatov za centre pa bomo podtaknili algoritmu za problem prilagodljivih atributov. Na ta način pridemo do naslednjega algoritma.

[3] $d(v, C) = \min_{c \in C} d(v, c)$.

Algoritem 5.19: Prilagodljivostni Gonzalezov algoritem

Vhod: Naloga (G, I, C).
Izhod: Zaporedje $(C_1, C_2, \ldots, C_n)$.

1. forall $i \in V$ do
2. $\quad C_i := \{c_i\}$, kjer je $c_i \in V_i$ poljuben;
3. while $|C_i| < p$ do
4. $\quad$ forall $i \in V$ do
5. $\quad\quad A_i = \{u \in V_i/C_i | \min_{c \in C_i} d(c, u) = \max_{v \in V_i/C_i} \min_{c \in C_i} d(c, v)\}$;
6. $\quad$ Sestavi (G, I', C), kjer je $s_i \in I'$ opisan z A_i;
7. $\quad (a_1, a_2, \ldots, a_n) :=$ reši (G, I', C);
8. $\quad$ forall $i \in V$ do
9. $\quad\quad C_i := C_i \cup \{a_i\}$;
10. end;
11. return $(C_1, C_2, \ldots, C_n)$;

V vrstici 1 in 2 najprej incializiramo začetne množice C_i za vse $i \in V$. Zanka v vrsticah od 3 do 10 se izvede $(p-1)$-krat, pri tem se na vsakem koraku $|C_i|$ poveča za 1. Po končanem izvajanju zanke velja $|C_i| = p$ za vse $i \in V$. Na vsakem koraku zanke najprej (vrstici 4 in 5) za vse $i \in V$ izračunamo množico A_i vozlišč, ki so najbolj oddaljena od C_i. Iz teh nato sestavimo nalogo (G, I', C) za problem prilagodljivih atributov in jo rešimo. Rešitev $(a_1, a_2, \ldots, a_n)$ nato (vrstici 8 in 9) uporabimo za širitev C_i, kjer $i \in V$.

Naj bo $r = \max_{i \in V} |V_i|$. Glavna zanka se izvede v $p-1$ korakih. Za izračun A_i porabimo $O(pr)$ korakov, torej je časovna zahtevnost vrstic 4 in 5 reda $O(pnr)$. Celotna časovna zahtevnost algoritma je torej $O(p^2nr + kT(n, r))$, kjer je $T(n, r)$ časovna zahtevnost reševanja problema prilagodljivih atributov. V primeru, da je G drevo, velja $T(n, r) = O(n^2r^2)$, torej je celotna časovna zahtevnost $O(p^2nr + pn^2r^2)$. V primeru splošnega grafa G pa gre seveda za algoritem s superpolinomsko časovno zahtevnostjo.

5.5 Večkriterijski prilagodljivi k-center

5.5.1 Vpeljava prilagodljivosti

V tem razdelku bomo opisali primer vpeljave prilagodljivosti s pomočjo večkriterijske optimizacije. Definirali bomo prilagodljivostni problem razmeščanja centrov, v katerem bo kriterijska funkcija sestavljena tako iz celotne kakovosti kot iz celotne prilagodljivosti. V podrazdelkih 3.2.2 in 3.2.3 smo za obe opisali več načinov izračuna. Na tem mestu bomo uporabljali način, v katerem sta definirani kot vsota partikularnih kakovosti oz. kot vsota partikularnih prilagodljivosti. Kriterijska funkcija večkriterijskega prilagodljivostnega problema je torej definirana kot

$$F(S_0, S_1, \ldots, S_n) = \alpha z_{sum}(S_1, S_2, \ldots, S_n) + \beta f_{sum}(S_1, S_2, \ldots, S_n),$$

pri čemer sta α in β konstanti, s katerima lahko uravnavamo pomembnost kakovosti oz. prilagodljivosti. Zapišimo še definicijo problema.

Problem 5.29: NAJMANJŠI VEČKRITERIJSKI PRILAGODLJIVI p-CENTER

Naloga: Trojica (G, I, C), kjer je scenarij $s_i \in I$ opisan z omrežjem $G_i = (V_i, E_i)$, in parameter $p \leq \min_{1 \leq i \leq n} |V_i|$.
Dopustna rešitev: Zaporedje $(C_1, C_2, \ldots, C_n)$, kjer je $C_i \subseteq V_i$ in $|C_i| = p$.
Kakovost dopustne rešitve: $F(C_1, C_2, \ldots, C_n)$.
Cilj: Minimizacija.

Problem je očitno $\mathcal{NP}$-težek, ker gre v primeru enega scenarija za problem NAJMANJŠI p-CENTER. Če ne velja trikotniška neenakost za povezave vseh grafov G_i, kjer $i \in V$, je tudi neaproksimabilen. V nadaljevanju bomo predpostavili veljavnost trikotniške neenakosti. Poleg tega se bomo osredotočili na reševanje preprostejše različice, v kateri je graf prehodov scenarijev sestavljen le iz dveh med seboj povezanih scenarijev. (Glej sliko 5.3.)

Slika 5.3: Graf prehodov scenarijev.

Torej gre za minimizacijo funkcije

$$F(C_1, C_2) = \alpha z_1(C_1) + \alpha z_2(C_2) + \beta f_{12}(C_1, C_2),$$

kjer sta $z_1(C_1)$ in $z_2(C_2)$ kriterijski funkciji problema NAJMANJŠI p-CENTER, izračunani za nalogi G_1 oz. G_2. Potrebno je torej poiskati rešitvi za dva scenarija, tako da je vsota kakovosti obeh rešitev in njune partikularne prilagodljivosti najmanjša. Robustna različica takega problema je v literaturi že obdelana v [HP98, BGKS98], kjer je predstavljen 3-aproksimacijski algoritem. Na podoben način bomo v nadaljevanju skonstruirali prav tako 3-aproksimacijski algoritem za večkriterijsko prilagodljivostno različico.

5.5.2 Aproksimabilnost osnovnega problema

Za problem NAJMANJŠI p-CENTER smo v predhodnem razdelku opisali Gonzalezov 2-aproksimacijski algoritem. Na tem mestu pa bomo za isti problem predstavili še enega izmed načinov reševanja, ki ravno tako vodi do 2-aproksimacijskega algoritma. Na podlagi tega bomo zgradili 3-aproksimacijski algoritem za NAJMANJŠI VEČKRITERIJSKI PRILAGODLJIVI p-CENTER. V nadaljevanju tega podrazdelka pojme in rezultate črpamo predvsem iz [BGKS98, HP98, HS85, Vaz01].

Naj bo $G = (V, E)$ graf. Množica $D \subseteq V$, za katero velja, da je vsako vozlišče $v \in V - D$ neposredno povezano z vsaj enim vozliščem v D, se imenuje *dominantna množica* vozlišč grafa G. Vprašanje, ali v danem grafu obstaja dominantna množica

moči kvečjemu k, je $\mathcal{NP}$-poln problem [GJ79]. S tem je povezan optimizacijski problem iskanja dominantne množice najmanjše moči[4].

Množica $I \subseteq V$, v kateri noben par vozlišč ni neposredno povezan v grafu G, se imenuje *neodvisna množica* vozlišč grafa G. Neodvisna množica, ki ni podmnožica kake druge neodvisne množice, se imenuje *maksimalna neodvisna množica*. To pomeni, da v I ne moremo dodati nobenega vozlišča iz V, da bi I še vedno ostala neodvisna. Maksimalno neodvisno množico I grafa G lahko sestavimo v polinomskem času, kar naredimo tako, da v I dodajamo vozlišča iz V/I dokler lahko, pri čemer na vsakem koraku v I dodamo poljubno vozlišče iz V/I, ki ni povezano s kakim od vozlišč v I. (Največja neodvisna množica pa je $\mathcal{NP}$-težek problem.)

Naj bo $G = (V, E)$ omrežje, kjer je $w(e)$ cena povezave $e \in E$. Graf $G(w) = (V, E')$, kjer $E' = \{e \in E | w(e) \leq w\}$, je *okleščeni graf*. Okleščeni graf $G(w)$ torej vsebuje vse povezave s ceno kvečjemu w originalnega grafa G. Vsak graf G vsebuje največ $|E| + 1$ različnih okleščenih grafov.

Reševanje problema Najmanjši p-center je enakovredno iskanju najmanjše vrednosti w, pri kateri okleščeni graf $G(w)$ vsebuje dominantno množico moči največ k [Mih04a, Vaz01]. V tem primeru so vsa vozlišča $v \in V$ povezana s povezavo dolžine kvečjemu w z nekim vozliščem iz D. Množica D je optimalna rešitev problema Najmanjši p-center. Takšni tehniki iskanja vrednosti w pravimo *parametrično kleščenje*[5]. Žal je iskanje najmanjše dominantne množice $\mathcal{NP}$-težek problem.

Kvadrat grafa G je graf $G^2 = (V, E^2)$, kjer $(u, v) \in E^2$, če v G med u in v obstaja pot, ki vsebuje kvečjemu dve povezavi. Iz G dobimo G^2 tako, da za vsako pot v G sestavljeno iz dveh povezav, dodamo v G^2 novo povezavo, katere dolžina je seštevek obeh. Velja naslednje: če je G graf, v katerem ima najdaljša povezava dolžino w, ima v G^2 najdaljša povezava dolžino največ $2w$. Ta lastnost je osnova za naslednjo lemo.

Lema 5.55 ▶ *Naj bo $G = (V, E)$ omrežje in naj bo I maksimalna neodvisna množica na grafu $G^2(w)$. Velja $z(I) \leq 2w$.*

Dokaz. V $G^2(w)$ so vsa vozlišča iz V/I neposredno povezana z vsaj enim vozliščem iz I. Če to ne bi veljalo, I ne bi bila maksimalna neodvisna množica. Potemtakem je I tudi dominantna množica v $G^2(w)$. Cena najdaljše povezave v $G^2(w)$ je kvečjemu $2w$. Vsako vozlišče v G je torej iz I dosegljivo po največ eni povezavi iz $G^2(w)$. □

Pravkar zapisana lema nam pove, da je maksimalna neodvisna množica I v G^2 pravzaprav 2-aproksimacijska rešitev problema Najmanjši p-center, kjer $p = |I|$. Z naslednjo lemo pa opišemo še odnos med močjo neodvisne in dominantne množice.

Lema 5.56 ▶ *Če je D dominantna množica v G in I neodvisna množica v grafu G^2, potem velja $|I| \leq |D|$.*

[4] Tej moči pogosto pravimo tudi dominantno število.

[5] *angl. parametric prunning*

Dokaz. Naj bo D najmanjša dominantna množica v G. Graf G je potemtakem sestavljen iz $|D|$ zvezd, vpetih v G, katerih središča so vozlišča iz D. V G^2 so te zvezde klike. Torej je v I kvečjemu eno vozlišče iz vsake klike. □

Sedaj lahko opišemo Hochbaum-Shmoysov 2-aproksimacijski algoritem za NAJMANJŠI p-CENTER. Zaporedoma preiskujemo okleščene grafe $G(w)$, kjer w narašča od 0 do vrednosti najdaljše povezave v G. Na vsakem koraku poiščemo maksimalno neodvisno množico I na grafu $G^2(w)$. Če $|I| > p$, potem zaradi leme 5.56 velja, da v G ne obstaja dominantna množica moči $\leq p$ in s tem tudi ne obstaja množica centrov C moči $\leq p$, kjer $z(C) \leq w$. V tem primeru nadaljujemo z naslednjim korakom. V nasprotnem primeru, t.j. $|I| \leq p$, pa I vrnemo kot rešitev, za katero zaradi leme 5.55 velja $z(I) \leq 2w$. Zaporedni postopek zagotavlja, da je w najmanjši, torej velja $w \leq z^*$. Postopek je zapisan kot naslednji algoritem.

Algoritem 5.20: Hochbaum-Shmoysov algoritem

Vhod: Omrežje $G = (V, E)$ in parameter $p \in \mathbb{Z}^+$.
Izhod: Množica centrov $C \subseteq V$.

```
1.  for w := w_0 to w_m do
2.      I := MaksimalnaNeodvisnaMnožica(G^2(w));
3.      if |I| ≤ p return I;
4.  end;
```

Pri tem predpostavljamo, da so uteži $w_0 = 0, w_1, w_2, \ldots, w_m$ povezav $e \in E$ pred zagonom algoritma urejene v nepadajočem vrstnem redu. Zanka (vrstica 1) poteka po vseh utežeh, pri čemer na vsakem koraku (vrstica 2) poiščemo maksimalno neodvisno množico na $G^2(w)$. Če je $I \leq p$, potem I vrnemo kot rešitev. Če Hochbaum-Shmoysov algoritem implementiramo z dvojiškim iskanjem, je njegova časovna zahtevnost $O(n^2 \log n)$.

5.5.3 Aproksimacijski algoritem

Ideja algoritma

Naj bosta $G_1 = (V_1, E_1)$ in $G_2 = (V_2, E_2)$ dve omrežji in naj bosta w_1 in w_2 ceni dveh poljubnih povezav v G_1 oz. G_2, t.j. $w_1 \in \{w(e) | e \in E_1\}$ in $w_2 \in \{w(e) | e \in E_2\}$. Naj bosta $S^*_{w_1} \subseteq V_1$ in $S^*_{w_2} \subseteq V_2$ partikularni rešitvi s kakovostjo w_1 oz. w_2, t.j. $z_1(S^*_{w_1}) = w_1$ in $z_2(S^*_{w_1}) = w_2$, za kateri velja, da je cena prilagajanja $f_{12}(S^*_{w_1}, S^*_{w_2})$ najmanjša. Torej velja

$$F(S^*_{w_1}, S^*_{w_2}) = w_1 + w_2 + f_{12}(S^*_{w_1}, S^*_{w_2}).$$

Ideja algoritma je, da poiščemo takšna $S_{w_1} \subseteq V_1$ in $S_{w_2} \subseteq V_2$, da velja $|S_{w_1}| \leq |S^*_{w_1}|$ in $|S_{w_2}| \leq |S^*_{w_2}|$ ter $z_1(S_{w_1}) \leq 3w_1$ in $z_2(S_{w_2}) \leq 3w_2$, pri čemer je njuna cena

prilagajanja manjša od prilagajanja $S^*_{w_1}$ in $S^*_{w_2}$, t.j.

$$f_{12}(S_{w_1}, S_{w_2}) \leq f_{12}(S^*_{w_1}, S^*_{w_2}). \tag{5.1}$$

Nadalje

$$\begin{aligned} F(S_{w_1}, S_{w_2}) &= z_1(S_{w_1}) + z_2(S_{w_2}) + f_{12}(S_{w_1}, S_{w_2}) \leq \\ &\leq 3w_1 + 3w_2 + f_{12}(S^*_{w_1}, S^*_{w_2}) \leq \\ &\leq 3F(S^*_{w_1}, S^*_{w_2}). \end{aligned} \tag{5.2}$$

Na ta način lahko sestavimo 3-aproksimacijski algoritem za Najmanjši Večkriterijski Prilagodljivi p-Center. Sedaj pa pokažimo, kako ustvarimo takšni S_{w_1} in S_{w_2}.

Konstrukcija S_{w_1} in S_{w_2}

Najprej v $G_1^2(w_1)$ in $G_2^2(w_2)$ poiščemo maksimalni neodvisni množici I_{w_1} oz. I_{w_2}. Ker $z_1(S^*_{w_1}) = w_1$, je $S^*_{w_1}$ dominantna množica v $G_1(w_1)$; enako velja za $S^*_{w_1}$. Zaradi leme 5.56 torej velja $|I_{w_1}| \leq |S^*_{w_1}|$ in $|I_{w_2}| \leq |S^*_{w_2}|$. Predpostavimo $|I_{w_1}| = |I_{w_2}|$, kasneje bomo razpravo razširili še na $|I_{w_1}| \neq |I_{w_2}|$. In nadalje zaradi leme 5.55 velja $z_1(I_{w_1}) \leq 2w_1$ in $z_2(I_{w_2}) \leq 2w_2$. Zapišimo in dokažimo naslednjo lemo [HP98, BGKS98].

Lema 5.57 ▶ *Naj bo $G = (V, E)$ omrežje in I maksimalna neodvisna množica v grafu $G^2(w)$. Naj bo S množica, v kateri vsak $v \in I$ nadomestimo s poljubnim $u \in \mathcal{N}_{G(w)}[v]$. Potem velja $z(S) \leq 3w$.*

Dokaz. Zaradi leme 5.55 velja $z(I) \leq 2w$, t.j. za vse $u \in V$ obstaja $v \in I$, da $d(u, v) \leq 2w$. Poleg tega za vse $x \in \mathcal{N}_{G(w)}[v]$, kjer $v \in I$, velja $d(x, v) \leq w$. Torej za vse $u \in V$ obstaja $x \in \mathcal{N}_{G(w)}[v]$, da $d(u, x) \leq 3w$. □

Naj bo torej $S_{w_1} \subseteq V_i$ množica, konstruirana iz I_{w_1}, tako da vsak $v \in I_{w_1}$ nadomestimo z nekim $u \in \mathcal{N}_{G_1(w_1)}[v]$. Enako konstruirajmo še S_{w_2} iz I_{w_2}, tako da vsak $v \in I_{w_2}$ nadomestimo z nekim $u \in \mathcal{N}_{G_2(w_2)}[v]$. Velja $|S_{w_1}| = |I_{w_1}| \leq |S^*_{w_1}|$ in $|S_{w_2}| = |I_{w_2}| \leq |S^*_{w_2}|$ ter $z_1(S_{w_1}) \leq 3w_1$ in $z_2(S_{w_2}) \leq 3w_2$.

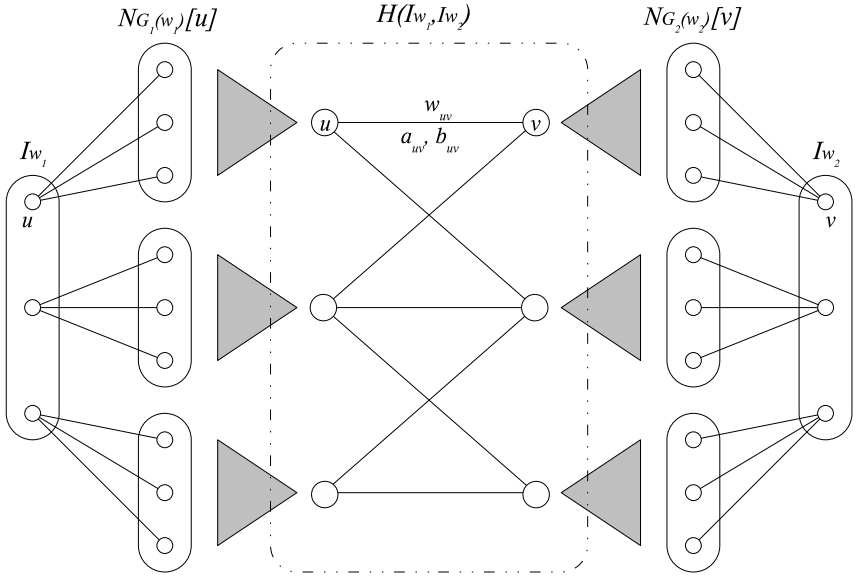
Pri konstrukciji S_{w_1} in S_{w_2} je potrebno izbirati takšna vozlišča iz $\mathcal{N}_{G_1(w_1)}[v]$ oz. $\mathcal{N}_{G_1(w_1)}[v]$, da bo veljala enačba 5.1. To naredimo s pomočjo problema Najmanjša Dodelitev. Natančneje, iz I_{w_1} in I_{w_2} ustvarimo poln dvodelen graf $H = (I_{w_1} \cup I_{w_2}, I_{w_1} \times I_{w_2})$. Za $(u, v) \in I_{w_1} \times I_{w_2}$ naj velja

$$w_{uv} = \min_{x \in \mathcal{N}_{G_1(w_1)}[u]} \min_{y \in \mathcal{N}_{G_2(w_2)}[v]} c_{12}(x, y). \tag{5.3}$$

Poleg tega si za vse (u, v) zapomnimo še

$$(a_{uv}, b_{uv}) = \arg \min_{x \in \mathcal{N}_{G_1(w_1)}[u]} \arg \min_{y \in \mathcal{N}_{G_2(w_2)}[v]} c_{12}(x, y),$$

Slika 5.4: Izračun S_{w_1} in S_{w_2}.



t.j. $a_{uv} \in \mathcal{N}_{G_1(w_1)}[u]$ in $b_{uv} \in \mathcal{N}_{G_2(w_2)}[v]$ sta vozlišči, ki zadoščata minimumu v enačbi 5.3. Konstrukcija H je prikazana na sliki 5.4.

Za tako ustvarjene soseščine velja naslednja lema.

Lema 5.58 ▶ *Naj bo $G = (V, E)$ graf in I maksimalna neodvisna množica na grafu G^2. Množice $\mathcal{N}[v]$, kjer $v \in I$, so med seboj paroma disjunktne.*

Dokaz. Naj bosta $u, v \in I$ dve različni vozlišči. Skupno vozlišče x v obeh soseščinah, t.j. $x \in \mathcal{N}[u]$ in $x \in \mathcal{N}[v]$, bi pomenilo obstoj poti med u in v, ki je sestavljena iz dveh povezav (preko x). To pa je v protislovju s trditvijo, da je I neodvisna množica v G^2. □

Iz leme seveda sledi, da so si vse $\mathcal{N}_{G_1(w_1)}[u]$, kjer $u \in I_{w_1}$, disjunktne; enako velja za vse $\mathcal{N}_{G_2(w_2)}[u]$, kjer $u \in I_{w_2}$. Naj bo M rešitev problema Najmanjša Dodelitev na tako konstruiranem omrežju H. Iz M skonstruiramo rešitvi S_{w_1} in S_{w_2} na naslednji način.

$$S_{w_1} = \{a_{uv} | (u, v) \in M\}$$

in

$$S_{w_2} = \{b_{uv} | (u, v) \in M\}.$$

Za tako konstruirani S_{w_1} in S_{w_2} velja naslednja lema.

Lema 5.59 ▶

$$f_{12}(S_{w_1}, S_{w_2}) \leq f_{12}(S^*_{w_1}, S^*_{w_2}).$$

Dokaz. Dovolj je, da pokažemo, da $S^*_{w_1}$ in $S^*_{w_2}$ predstavljata neko dopustno ujemanje v H. Optimalnost $f_{12}(S_{w_1}, S_{w_2})$ nato omogoča zaključek dokaza.

Vsaka od $\mathcal{N}_{G_1(w_1)}[u]$, kjer $u \in I_{w_1}$, vsebuje vsaj eno vozlišče iz $S^*_{w_1}$, in enako vsaka od $\mathcal{N}_{G_2(w_2)}[u]$, kjer $u \in I_{w_2}$, vsebuje vsaj eno vozlišče iz $S^*_{w_2}$. Pokažimo to za $S^*_{w_1}$, kar zaradi simetrije velja tudi za $S^*_{w_2}$. Ker $z_1(S^*_{w_1}) = w_1$, je $S^*_{w_1}$ dominantna množica v $G_1(w_1)$. Torej za vsak $u \in I_{w_1}$ velja bodisi $u \in S^*_{w_1}$ bodisi v $G_1(w_1)$ obstaja povezava (u, v), kjer $v \in S^*_{w_1}$. Torej obstaja $v \in S^*_{w_1}$, tako da za vsak $u \in I_{w_1}$ velja $v \in \mathcal{N}_{G_1(w_1)}[u]$.

Naj bo M^* najcenejše ujemanje v $H^* = (S^*_{w_1} \cup S^*_{w_2}, S^*_{w_1} \times S^*_{w_2})$. Velja torej $f_{12}(S^*_{w_1}, S^*_{w_2}) = \sum_{e \in M^*} w(e)$. Zaradi leme 5.58 so si $\mathcal{N}_{G_1(w_1)}[u]$, kjer $u \in I_{w_1}$, disjunktne.

Potemtakem v primeru, da $|I_{w_1}| = |S^*_{w_1}|$, vsaka od $\mathcal{N}_{G_1(w_1)}[u]$, kjer $u \in I_{w_1}$, vsebuje natanko eno vozlišče iz $S^*_{w_1}$. Uteži povezav v H so torej prav gotovo manjše ali enake od uteži povezav v H^*. Torej najcenejše ujemanje v H prav gotovo ni dražje od M^*.

Če pa $|I_{w_1}| < |S^*_{w_1}|$, potem lahko katera od $\mathcal{N}_{G_1(w_1)}[u]$, kjer $u \in I_{w_1}$, vsebuje več kot eno vozlišče iz $S^*_{w_1}$. Lahko tudi obstaja vozlišče v $S^*_{w_1}$, ki ni v nobeni od $\mathcal{N}_{G_1(w_1)}[u]$, kjer $u \in I_{w_1}$. To pa ne predstavlja nobene težave, v obeh primerih najcenejše ujemanje v H ni dražje od M^*. □

Z enačbo 5.2 smo pravzaprav že dokazali naslednjo trditev.

Trditev 5.60 ▶

$$F(S_{w_1}, S_{w_2}) \leq 3F(S^*_{w_1}, S^*_{w_2}).$$

Oglejmo si, kako lahko opisano konstrukcijo razširimo še na $|I_{w_1}| \neq |I_{w_2}|$. Naj velja npr. $|I_{w_1}| < |I_{w_2}|$. V tem primeru v I_{w_1} lahko dodamo $|I_{w_2}| - |I_{w_1}|$ pomožnih vozlišč, tako da velja $|I_{w_1}| = |I_{w_2}|$. Cena prilagajanja pomožnega vozlišča iz I_{w_1} v neko vozlišče $v \in I_{w_2}$ pa predstavlja ceno postavitve centra v vozlišču v. Analogno lahko postopamo v primeru $|I_{w_1}| > |I_{w_2}|$.

Algoritem

V vsakem od grafov G_1 in G_2 je m povezav, torej je toliko tudi različnih w_1 oz. w_2. Preveriti je torej potrebno vse možne pare $w_1 \in \{w(e)|e \in E_1\}$ in $w_2 \in \{w(e)|e \in E_2\}$, t.j. $O(m^2)$ parov. Za vsak par (w_1, w_2) najprej izračunamo I_{w_1} in I_{w_2}.

Če $|I_{w_1}| > p$, potem zaradi leme 5.56 v $G_1(w_1)$ ne obstaja $S^*_{w_1} \subseteq V_1$, tako da $|S^*_{w_1}| \leq p$ in $z_1(S^*_{w_1}) = w_1$. ($S^*_{w_1}$ je dominantna množica v $G_1(w_1)$.) Enako velja v primeru $|I_{w_2}| > p$. Če torej $|I_{w_1}| \leq p$ in $|I_{w_2}| \leq p$, potem lahko uporabimo zgoraj opisano konstrukcijo za S_{w_1} in S_{w_2}. Izmed vseh tako konstruiranih S_{w_1} in S_{w_2} za vse (w_1, w_2) pa izberemo takšni z najmanjšim $F(S_{w_1}, S_{w_2})$. Postopek je formalno zapisan kot naslednji algoritem.

Algoritem 5.21: Večkriterijsko prilagodljivostno razmeščanje centrov

Vhod: Naloga (G, I, C), kjer sta G_1 in G_2 omrežji.
Izhod: Rešitev (S_1, S_2).

1. forall $(w_1, w_2) \in \{w(e)|e \in E_1\} \times \{w(e)|e \in E_2\}$ do
2. $I_1 :=$ MaksimalnaNeodvisnaMnožica$(G_1^2(w))$;
3. $I_2 :=$ MaksimalnaNeodvisnaMnožica$(G_2^2(w))$;
4. if $|I_1| \leq p$ and $|I_2| \leq p$ then
5. Konstruiraj $H = (I_1 \cup I_2, I_1 \times I_2)$;
6. $M =$ reši H;
7. Konstruiraj S_1 in S_2 iz M;
8. end;
9. return (S_1, S_2) z najmanjšim $F(S_1, S_2)$;

Zanka v algoritmu se izvede za vse (w_1, w_2), torej porabi $O(m^2)$ časa, kjer je $m = \max(|E_1|, |E_2|)$. Množici I_1 in I_2 lahko poiščemo v času $O(mn)$, kjer je $n = |V_1| = |V_2|$. Konstrukcija omrežja H zahteva $O(m)$ časa. Iskanje ujemanja M v H pa $O(p^3)$. Medtem ko lahko S_1 in S_2 konstruiramo neposredno iz M, t.j. v času $O(n)$. Časovna zahtevnost algoritma je torej $O(m^2(mn + p^3))$.

Poglavje 6

Zaključek

Le drzne spekulacije, ne pa kopičenje dejstev,
nas lahko popeljejo naprej.
– Albert Einstein

Zaključno poglavje podaja povzetek avtorjevih izvirnih prispevkov k znanosti. Poleg tega naštejemo še nekatere možnosti za nadaljevanje dela.

6.1 Povzetek

Glavne prispevke znanosti lahko razdelimo na naslednje tri večje sklope. V prvem smo formalizirali prilagodljivost v optimizacijskih problemih. Prilagodljivost in z njo povezane pojme, kot so graf prehodov scenarijev, partikularna kakovost, partikularna prilagodljivost itd., smo natančno definirali. Predstavili smo tudi več možnih načinov za vpeljavo prilagodljivosti v optimizacijske probleme. Opisali smo praktično uporabnost prilagodljivosti, kot je npr. obvladovanje nedoločenosti v vhodnih podatkih optimizacijskih problemov.

V drugem sklopu smo obdelali posebne vrste prilagodljivost, ki smo jo poimenovali problemi prilagodljivih atributov. Pokazali smo računsko zahtevnost, torej $\mathcal{NP}$-težkost in neaproksimabilnost teh problemov. Zato smo predstavili več poenostavljenih različic, ki so "lažje" za reševanje. Tako smo zgradili natančne in približne postopke za reševanje različnih vrst problemov prilagodljivih atributov.

V tretjem sklopu smo vpeljali prilagodljivosti v nekatere že znane optimizacijske probleme. Na ta način smo želeli prikazati splošnost prilagodljivosti. Prilagodljivost smo vpeljali v več problemov razmeščanja centrov ter v problem iskanja najmanjšega vpetega drevesa.

Skozi celotno disertacijo smo področja obdelovali predvsem z vidika teorije računske zahtevnosti, analize algoritmov, teorije grafov, problemov pretokov in ujemanj itd.

6.2 Prispevki k znanosti

Poleg nekaterih že znanih rezultatov iz literature so v doktorski disertaciji predstavljeni naslednji avtorjevi izvirni prispevki k znanosti.

Prilagodljivost v optimizacijskih problemih.
V tretjem poglavju smo prilagodljivost najprej predstavili na splošno in jo primerjali s sorodnimi pristopi, ki se pojavljajo v literaturi. Nato smo prilagodljivost v optimizacijskih problemih natančno razdelali, opredelili in definirali. Opisali smo več vrst prilagodljivosti, od katerih sta najpomembnejši sum in max prilagodljivost.

Nedoločenost vhodnih podatkov.
V razdelku 2.1 smo obdelali različne načine matematičnega opisa oz. predstavitve vhodnih podatkov. Na podlagi tega smo v razdelku 2.4 opisali različne vrste nedoločenosti vhodnih podatkov v optimizacijskih problemih. Prikazali smo nekatere načine za modeliranje nedoločenosti, med katerimi smo izpostavili predvsem modeliranje nedoločenosti s scenariji. Poleg tega smo predstavili prilagodljivost kot metodo obvladovanja nedoločenosti vhodnih podatkov.

Problemi prilagodljivih atributov.
V četrtem poglavju smo vpeljali novo vrsto problemov, imenovanih problemi prilagodljivih atributov, v katerih gre za enostavno obliko prilagodljivosti. Glede na sum in max prilagodljivost smo definirali (razdelek 4.1) problema Najmanjša $\sum$-Prilagodljivost p-Atributov in Najmanjša max-Prilagodljivost p-Atributov, ki predstavljata najsplošnejši različici v skupini problemov prilagodljivih atributov.

Zahtevnost reševanja problemov prilagodljivih atributov.
V razdelku 4.2 smo analizirali zahtevnost reševanja problemov prilagodljivih atributov. Za obe splošni različici kakor tudi za več poenostavitev smo pokazali $\mathcal{NP}$-težkost in analizirali njihovo aproksimabilnost. Za nekatere različice smo predstavili natančne polinomske algoritme, za druge pa hevristične ali aproksimacijske algoritme.

Algoritem za problema prilagodljivih atributov na drevesu.
V razdelku 4.4 smo predstavili polinomski natančni algoritem za problema Najmanjša $\sum$-Prilagodljivost 1-Atributov in Najmanjša max-Prilagodljivost 1-Atributov, v katerih je graf prehodov scenarijev drevo. Analizirali smo časovno zahtevnost algoritma in dokazali pravilnost in optimalnost predstavljenega algoritma.

Izločanje scenarijev z osamljenimi atributi.
Predstavili smo učinkovit način, ki omogoča zmanjševanje velikosti naloge v primeru, da scenariji vsebujejo osamljene atribute. Način izločanja scenarijev

z osamljenimi atributi smo v razdelku 4.5 natančno opisali in formalno dokazali njegovo pravilnost.

Algoritem za problem prilagodljivih atributov na splošnem grafu.
Na podlagi izločanja scenarijev z osamljenimi atributi smo zasnovali (razdelek 4.5) algoritem za natančno reševanje problemov Najmanjša $\sum$-Prilagodljivost 1-Atributov in Najmanjša max-Prilagodljivost 1-Atributov. Pokazali smo pravilnost algoritma ter analizirali njegovo časovno zahtevnost.

Dvostopenjski problemi investiranja.
V razdelku 4.6 smo obdelali več različic problemov prilagodljivih atributov, v katerih je graf prehodov scenarijev dvostopenjski. Za primer $n = 3$ scenarijev smo tako razvili polinomski natančni algoritem za Najmanjše 2-Stopenjsko $\sum$-Prilagodljivo Investiranje in polinomski 2-aproksimacijski algoritem za Najmanjše 2-Stopenjsko max-Prilagodljivo Investiranje. Za različice, v katerih $n \geq 4$, pa smo pokazali $\mathcal{NP}$-težkost in neaproksimabilnost.

Verižno prilagajanje.
Za problem Najmanjša $\sum$-Prilagodljivost p-Atributov, v katerem je graf prehodov scenarijev veriga, smo razvili (razdelek 4.7) polinomski natančni algoritem. Pri tem smo uporabili že znane rezultate s področja pretokov v omrežju. Za algoritem smo pokazali polinomsko časovno zahtevnost in dokazali njegovo pravilnost in optimalnost. Algoritem je hkrati tudi $|E|$-aproksimacijski algoritem za problem Najmanjša max-Prilagodljivost p-Atributov, kjer je graf prehodov scenarijev veriga.

Problemi splošne prilagodljivosti.
V tretjem poglavju smo predstavili metodo, s katero lahko prilagodljivost preprosto vpeljemo v nekatere že znane optimizacijske probleme. Poleg tega smo metodo uporabili na nekaterih konkretnih primerih problemov, ki smo jih opisali v petem poglavju.

Prilagodljivi 1-center.
V problem Najmanjši 1-Center smo vpeljali (razdelek 5.2) prilagodljivost in s tem definirali problema Najmanjši $\sum$-Prilagodljivi 1-Center in Najmanjši max-Prilagodljivi 1-Center. Pokazali smo $\mathcal{NP}$-težkost obeh problemov. Predstavili smo algoritem za njuno natančno reševanje, ki ima v primeru drevesnega grafa prehodov scenarijev polinomsko časovno zahtevnost. Algoritem smo izpeljali s pomočjo algoritmov za problem prilagodljivih atributov.

Prilagodljivo najmanjše vpeto drevo.
V problem Najmanjše Vpeto Drevo smo vpeljali prilagodljivost (razdelek

5.3) in s tem definirali problema NAJMANJŠE $\sum$-PRILAGODLJIVO VPETO DREVO in NAJMANJŠE max-PRILAGODLJIVO VPETO DREVO. Pokazali smo $\mathcal{NP}$-težkost obeh problemov. Predstavili smo algoritem za njuno hevristično reševanje, ki ima polinomsko časovno zahtevnost v primeru drevesnega grafa prehodov scenarijev. Algoritem smo izpeljali s pomočjo algoritma za problem prilagodljivih atributov.

Prilagodljivo razmeščanje centrov.
V razdelku 5.4 smo vpeljali prilagodljivost v problem NAJMANJŠI p-CENTER in s tem definirali problema NAJMANJŠI $\sum$-PRILAGODLJIVI p-CENTER in NAJMANJŠI max-PRILAGODLJIVI p-CENTER. Za oba problema smo pokazali $\mathcal{NP}$-težkost. Poleg tega smo predstavili algoritem za njuno hevristično reševanje, ki smo ga izpeljali s pomočjo problemov prilagodljivih atributov.

Prilagodljivi večkriterijski p-center.
V razdelku 5.5 smo definirali večkriterijsko različico prilagodljivostnega problema razmeščanja centrov, imenovan NAJMANJŠI VEČKRITERIJSKI PRILAGODLJIVI p-CENTER. V kriterijski funkciji nastopata tako partikularna kakovost kot partikularna prilagodljivost. Za različico problema, v kateri nastopata dva scenarija, smo izpeljali 3-aproksimacijski algoritem in dokazali njegovo pravilnost.

6.3 Nadaljevanje dela

Prilagodljivost v optimizacijskih problemih je novo in zelo obširno področje, zato so tudi možnosti za nadaljnje raziskave številne. Omenimo nekatere.

Partikularna prilagodljivost.
Za izračun partikularne prilagodljivosti $f_{ij}(S_i, S_j)$ smo predpostavljali, da je moč partikularnih rešitev S_i in S_j enaka. Potrebno bi bilo raziskati še primer, ko $|S_i| \neq |S_j|$. Zanimivo bi bilo raziskati tudi drugačne definicije partikularne prilagodljivosti, kot smo jo podali v podrazdelku 4.1.2. Tako bi uporabnost prilagodljivosti razširili, ker bi jo lahko vpeljali v še več optimizacijskih problemov.

Celotna kakovost.
Predstavili smo tri načine izračuna celotne prilagodljivosti iz partikularne prilagodljivosti. Zanimiva bi bila tudi študija drugačnih izračunov, kot je npr. vsota ali maksimum partikularnega obžalovanja, kjer je partikularno obžalovanje definirano podobno kot v problemih robustnosti.

Vpliv grafa prehodov scenarijev.
V disertaciji smo pokazali nekaj posebnih primerov grafa prehodov scenarijev, ki zmanjšajo zahtevnost reševanja prilagodljivostnih problemov. Zanimivo bi

bilo raziskati še drugačne primere grafov, med njimi prav gotovo cikel, mrežo, krožni graf itd.

Hevristični algoritem za prilagodljivost p atributov na splošnem grafu.
Za reševanje problemov prilagodljivih p atributov nimamo na voljo hevrističnega algoritma. Kljub temu smo za reševanje splošnih prilagodljivostnih problemov uporabljali algoritem, ki p-krat izvede algoritem za problem prilagodljivih $p = 1$ atributov. Potrebno bi bilo analizirati slednjega in / ali raziskati alternativne postopke za reševanje prilagodljivosti p atributov na splošnem grafu.

Max prilagodljivost atributov na verigi.
Predstavili smo polinomski algoritem za natančno reševanje problema sum prilagodljivosti atributov na verižnem grafu prehodov scenarijev. Smiselno bi bilo raziskati še max prilagodljvost na enakem grafu. Ali gre za $\mathcal{NP}$-težek problem? Odgovor na to je verjetno da, kar bi morda lahko dokazali na podoben način kot $\mathcal{NP}$-težkost na dvostopenjskem grafu, kjer $n \geq 4$. Poraja se tudi vprašanje, ali v primeru $\mathcal{NP}$-težkosti problema obstaja aproksimacijski algoritem.

Prilagodljivost in modeliranje nedoločenosti.
Dobro bi bilo raziskati možnost vpeljave prilagodljivosti v probleme, v katerih je nedoločenost modelirana na drugačen način kot s scenariji.

Izločanje vozlišč.
Predstavili smo algoritem za izločanje vozlišč iz grafa. Zanimivo bi bilo natančneje analizirati povprečno število delov, na katere razpade nek graf ob izločitvi poljubnega vozlišča. Takšno analizo bi lahko uporabili za natančnejšo analizo časovne zahtevnosti algoritma 4.12.

Pri opisu omenjenega algoritma nismo povedali ničesar o načinu izbire vozlišča za izločanje. Prav gotovo obstajajo načini izločanja, ki hitreje vodijo do rešitve kot poljubno izbiranje. Nekateri načini so verjetno bolj primerni za neko vrsto grafov drugi za druge. Zanimivo bi bilo analizirati te načine izločanja.

Vpeljava prilagodljivosti v znane optimizacijske probleme.
Optimizacijskih problemov je ogromno. Zanimivo bi bilo vpeljati prilagodljivost še v druge že znane optimizacijske probleme, predvsem v tiste probleme, ki se pogosto pojavljajo v praksi.

Odnos med optimizacijskim problemom in prilagodljivostno različico.
Dobro bi bilo raziskati odnos med optimizacijskim problemom in njegovo prilagodljivostno različico glede zahtevnosti reševanja. Vsi prilagodljivostni problemi, obdelani v disertaciji, so $\mathcal{NP}$-težki na splošnem grafu. Vprašanje, ki ostaja odprto, je, ali sploh obstaja prilagodljivostni problem, ki ne bi bil $\mathcal{NP}$-težek na splošnem grafu prehodov scenarijev.

Dodatek A

Oznake

A.1 Splošne oznake

$G = (V, E)$	graf G z vozlišči V in povezavami $E \subseteq V \times V$
n	število vozlišč grafa, t.j. $n = \|V\|$
m	število povezav grafa, t.j. $m = \|E\|$
u, v, i, j	oznake vozlišč grafa
e, f	oznake povezav grafa
$w(e)$	utež povezave e
$\mathcal{N}(u)$	odprta soseščina grafa
$\mathcal{N}[u]$	zaprta soseščina grafa
$P = (I, S, m, cilj)$	optimizacijski problem P
I	množica nalog problema
x	naloga $x \in I$ problema
$S(x)$	funkcija $S(x)$, ki vrne množico dopustnih rešitev naloge x
y	rešitev $y \in S(x)$ problema
$m(x, y)$	kriterijska funkcija problema
$cilj$	cilj optimizacijskega problema
$D(x, y)$	absolutna napaka rešitve y naloge x
$E(x, y)$	relativna napaka rešitve y naloge x
$R(x, y)$	performančni kvocient rešitve y naloge x
A	algoritem
ρ	aproksimacijski faktor
f_{uv}	pretok skozi povezavo (u, v)
c_{uv}	kapaciteta povezave (u, v)
w_{uv}	cena povezave (u, v)
(f_{uv})	rešitev problema pretokov
$\|f_{uv}\|$	vrednost pretoka
M	ujemanje

A.2 Oznake v problemih prilagodljivosti

(G, I, C, p)	naloga prilagodljivih atributov
(G, I, C)	naloga prilagajanja ($p = 1$ atributov)
$G = (V, E)$	graf prehodov scenarijev
I	množica scenarijev
s_i	scenarij $s_i \in I$
$S(s_i)$	dopustne rešitve scenarija s_i
A_i	množica atributov, ki opisuje scenarij s_i
a_{ik}, a_{jl}	atributa $a_{ik} \in A_i$ in $a_{jl} \in A_j$.
$(a_1, a_2, \ldots, a_n)$	rešitev problema prilagajanja $p = 1$ atributov
$(S_1, S_2, \ldots, S_n)$	rešitev problema prilagajanja p atributov
C	množica funkcij cen prilagajanja atributov
c_{ij}	cena prilagajanja atributov
p	število iskanih atributov
q	najmanjše število atributov
r	največje število atributov
$z_i(S_i)$	kakovost rešitve S_i za scenarij s_i
$z(S_1, S_2, \ldots, S_n)$	celotna kakovost
α, α_i	faktor omejitve kakovosti
$f_{ij}(S_i, S_j)$	partikularna prilagodljivost partikularnih rešitev S_i in S_j
$f(S_1, S_2, \ldots, S_n)$	celotna prilagodljivost
$f_{sum}(S_1, S_2, \ldots, S_n)$	celotna sum prilagodljivost
$f_{max}(S_1, S_2, \ldots, S_n)$	celotna max prilagodljivost
$f_{min}(S_1, S_2, \ldots, S_n)$	celotna min prilagodljivost
$F(S_1, S_2, \ldots, S_n)$	večkriterijska funkcija

Dodatek B

Programi za GLPK

B.1 ILP-A

Na tem mestu podajamo izpis celoštevilskega linearnega programa ILP-A za problem Najmanjša $\sum$-Prilagodljivost 1-Atributov.

```
# število vozlišč grafa (scenarijev)
param n, integer, > 0;

# število atributov v scenariju
param na{1..n}, integer, > 0;

# matrika sosednosti za graf prehodov scenarijev
param e{i in 1..n, j in 1..n}, integer, >= 0, <= 1;

# cena prilagajanja atributa [i,k] v atribut [j,l]
param c{i in 1..n, k in 1..na[i], j in 1..n, l in 1..na[j]} :=
        (abs(i-j) + abs(k-l));

# x[i,k] = 1, če in samo če je atribut [i,k] izbran
var x{i in 1..n, 1..na[i]}, integer, >= 0, <= 1;

# y[i,k,j,l] = 1, če in samo če je atribut [i,k] prilagajan v [j,l]
var y{i in 1..n, 1..na[i], j in 1..n, 1..na[j]}, integer, >= 0, <= 1;

# omejitev za izbiro enega atributa
s.t. attr_cnt{i in 1..n}:
    sum{k in 1..na[i]} x[i,k] = 1;

# povezava med spremenljivkama x in y
s.t. edges1:
```

```
    sum{i in 1..n, k in 1..na[i], j in 1..n, l in 1..na[j]}
        y[i, k, j, l] = n^2;

s.t. edges2{i in 1..n, k in 1..na[i], j in 1..n, l in 1..na[j]}:
    x[i,k] + x[j,l] >= 2 * y[i,k,j,l];

# kriterijska funkcija
minimize obj: sum{i in 1..n, k in 1..na[i], j in 1..n, l in 1..na[j]}
    e[i,j] * c[i, k, j, l] * y[i, k, j, l];
```

B.2 ILP-B

Na tem mestu podajamo izpis celoštevilskega linearnega programa ILP-B za problem Najmanjša $\sum$-Prilagodljivost 1-Atributov.

```
# število vozlišč grafa (scenarijev)
param n, integer, > 0;

# število atributov v scenariju
param na{1..n}, integer, > 0;

# matrika sosednosti za graf prehodov scenarijev
param e{i in 1..n, j in 1..n}, integer, >= 0, <= 1;

# cena prilagajanja atributa [i,k] v atribut [j,l]
param c{i in 1..n, k in 1..na[i], j in 1..n, l in 1..na[j]} :=
        (abs(i-j) + abs(k-l));

# x[i,k] = 1, če in samo če je atribut [i,k] izbran
var x{i in 1..n, 1..na[i]}, integer, >= 0, <= 1;

# y[i,k,j,l] = 1, če in samo če je atribut [i,k] prilagajan v [j,l]
var y{i in 1..n, 1..na[i], j in 1..n, 1..na[j]}, integer, >= 0, <= 1;

# omejitev za izbiro enega atributa
s.t. attr_cnt{i in 1..n}:
    sum{k in 1..na[i]} x[i,k] = 1;

# povezava med spremenljivkama x in y
s.t. edges1{i in 1..n, k in 1..na[i]}:
    sum{j in 1..n, l in 1..na[j]} y[i, k, j, l] = n * x[i, k];
```

```
s.t. edges2{j in 1..n, l in 1..na[j]}:
    sum{i in 1..n, k in 1..na[i]} y[i, k, j, l] = n * x[j, l];

# kriterijska funkcija
minimize obj: sum{i in 1..n, k in 1..na[i], j in 1..n, l in 1..na[j]}
    e[i,j] * c[i, k, j, l] * y[i, k, j, l];
```

B.3 Testni podatki

Podajmo še primer testnih podatkov za programa ILP-A in ILP-B. V primeru velja $|A_i| = 4$ za vse $i \in V$.

```
data;

param n := 5;

param na := 1 4 2 4 3 4 4 4 5 4;

param e: 1 2 3 4 5 :=
      1  0 1 0 0 0
      2  0 0 0 1 0
      3  1 1 0 0 1
      4  0 0 1 0 0
      5  0 0 0 0 0;

end;
```

Dodatek C

Definicije problemov

C.1 Odločitveni problemi

Problem C.1: 3D-Ujemanje

Naloga: Množica $T \subseteq A \times B \times C$, pri čemer so A, B in C paroma disjunktne. Velja $|A| = |B| = |C| = p$.
Vprašanje: Ali obstaja 3D-ujemanje $T' \subseteq T$, kjer velja $|T'| = p$?

Problem C.2: Hamiltonov Cikel

Naloga: Graf $G = (V, E)$.
Vprašanje: Ali obstaja cikel $T = (v_{i_1}, v_{i_2}, \ldots, v_{i_n})$ v grafu G, da vsako vozlišče $v \in V$ obiščemo natanko enkrat?

Problem C.3: Pokritje Množice

Naloga: Množica $U = \{u_1, \ldots, u_n\}$ in družina $\mathcal{U} = \{U_1, U_2, \ldots, U_m\}$ podmnožic $U_i \subseteq U$, kjer velja $\cup_{X \in \mathcal{U}} X = U$, ter parameter $p \in \mathbb{N}$.
Vprašanje: Ali obstaja poddružina $T \subseteq \mathcal{U}$, kjer $|T| \leq p$, da $\cup_{X \in T} X = U$?

Problem C.4: Razdelitev

Naloga: Množica elementov $A = \{a_1, a_2, \ldots, a_p\}$, kjer elementu $a_i \in A$ pripada utež $w_i \in \mathbb{Z}^+$. Velja $\sum_i w_i = 2B$.
Vprašanje: Ali obstaja razdelitev $X, Y \subseteq A$, tako da velja $\sum_{a_i \in X} w_i = \sum_{a_i \in Y} w_i = B$?

Problem C.5: Trgovski Potnik

Naloga: Množica mest $\{c_1, c_2, \ldots, c_n\}$, matrika $D = (d_{ij})_{n \times n}$, kjer $d_{ij} \in \mathbb{Z}^+$ predstavlja razdaljo od mesta c_i do c_j in parameter $B \in \mathbb{Z}^+$.
Vprašanje: Ali obstaja permutacija $T = (c_{i_1}, c_{i_2}, \ldots, c_{i_n})$ mest, da je njena kakovost $m(T) = \sum_{k=1}^{n-1} d_{i_k, i_{k+1}} + d_{i_n, i_1}$ kvečjemu B, t.j. $m(T) \leq B$?

Problem C.6: Vozliščno pokritje

Naloga: Neusmerjen graf $G = (V, E)$ in parameter $B \in \mathbb{Z}^+$.
Vprašanje: Ali obstaja vozliščno pokritje $C \subseteq V$ grafa G, tako da $|C| \leq B$?

C.2 Optimizacijski problemi

Problem C.7: Največji Pretok

Naloga: Omrežje $G = (V, E)$, kjer je za vsako povezavo $(u, v) \in E$ podana kapaciteta $c_{uv} \geq 0$. Podana sta še izvor $s \in V$ in ponor $t \in V$.
Dopustna rešitev: Pretok (f_{uv}) iz s v t.
Kakovost dopustne rešitve: $|(f_{uv})|$.
Cilj: Maksimizacija.

Problem C.8: Najmanjša Cena Pretoka

Naloga: Omrežje $G = (V, E)$, kjer je za vsako povezavo $(u, v) \in E$ podana njena cena $w_{uv} \geq 0$ in kapaciteta $c_{uv} \geq 0$. Podani so še ciljna vrednost celotnega pretoka p, izvor $s \in V$ in ponor $t \in V$.
Dopustna rešitev: Pretok (f_{uv}) iz s v t z vrednostjo p.
Kakovost dopustne rešitve: $\sum_{(u,v) \in E} f_{uv} w_{uv}$.
Cilj: Minimizacija oz. odločitev, da rešitev ne obstaja.

Problem C.9: Največje Ujemanje

Naloga: Graf $G = (V, E)$.
Dopustna rešitev: Ujemanje $M, M \subseteq E$, v G.
Kakovost dopustne rešitve: $|M|$.
Cilj: Maksimizacija.

Problem C.10: Največja Cena Ujemanja

Naloga: Omrežje $G = (V, E)$, kjer je utež $w(e)$ cena povezave $e \in E$.
Dopustna rešitev: Ujemanje $M, M \subseteq E$, v G.
Kakovost dopustne rešitve: $\sum_{e \in M} w(e)$.
Cilj: Maksimizacija.

Problem C.11: Najmanjša Cena Popolnega Ujemanja

Naloga: Omrežje $G = (V, E)$, kjer je utež $w(e)$ cena povezave $e \in E$.
Dopustna rešitev: Popolno ujemanje $M, M \subseteq E$, v G.
Kakovost dopustne rešitve: Cena ujemanja M, t.j. $\sum_{e \in M} w(e)$.
Cilj: Minimizacija.

Problem C.12: Najmanjša Sprememba Grafa v Gozd

Naloga: Graf $G = (V, E)$.
Dopustna rešitev: Podmnožica $U \subseteq V$, tako da je graf $H(V/U, F)$ induciran nad G z vozlišči V/U gozd.
Kakovost dopustne rešitve: $|U|$.
Cilj: Minimizacija.

Problem C.13: Najmanjša Sprememba Grafa v Drevo

Naloga: Graf $G = (V, E)$.
Dopustna rešitev: Podmnožica $U \subseteq V$, tako da je graf $H(V/U, F)$ induciran nad G z vozlišči V/U drevo.
Kakovost dopustne rešitve: $|U|$.
Cilj: Minimizacija.

Problem C.14: Največji Induciran Podgraf z Lastnostjo p

Naloga: Graf $G = (V, E)$ in lastnost p.
Dopustna rešitev: Podmnožica $U \subseteq V$ tako, da ima graf $H(V/U, F)$ induciran nad G z vozlišči V/U lastnost p.
Kakovost dopustne rešitve: $|V/U|$.
Cilj: Maksimizacija.

Problem C.15: Najmanjše Vpeto Drevo

Naloga: Povezano omrežje $G = (V, E)$, kjer je $w(e)$ cena povezave $e \in E$.
Dopustna rešitev: Vpeto drevo $T = (V, F)$ v G.
Kakovost dopustne rešitve: $\sum_{e \in F} w(e)$.
Cilj: Minimizacija.

Problem C.16: Najmanjši 1-Center

Naloga: Neusmerjeno polno omrežje $G = (V, E)$, kjer je $d(u, v)$ dolžina povezave $(u, v) \in E$.
Dopustna rešitev: $c \in V$.
Kakovost dopustne rešitve: $z(c) = \max_{v \in V} d(c, v)$.
Cilj: Minimizacija.

Problem C.17: Najmanjši p-center

Naloga: Neusmerjeno polno omrežje $G = (V, E)$ in pozitivno celo število $p \leq |V|$.
Dopustna rešitev: Množica $C \subseteq V$, tako da $|C| \leq p$.
Kakovost dopustne rešitve: $z(S) = \max_{v \in V} \min_{c \in C} d(v, c)$.
Cilj: Minimizacija.

Problem C.18: Najmanjša 1-Mediana

Naloga: Neusmerjeno polno omrežje $G = (V, E)$, kjer je $d(u, v)$ dolžina povezave $(u, v) \in E$.
Dopustna rešitev: $c \in V$.
Kakovost dopustne rešitve: $z(c) = \sum_{v \in V} d(c, v)$.
Cilj: Minimizacija.

Problem C.19: Najmanjša k-mediana

Naloga: Neusmerjeno polno omrežje $G = (V, E)$ in pozitivno celo število $p \leq |V|$.
Dopustna rešitev: Množica $C \subseteq V$, tako da $|C| \leq p$.
Kakovost dopustne rešitve: $z(S) = \sum_{v \in V} \min_{c \in C} d(v, c)$.
Cilj: Minimizacija.

C.3 Problemi prilagodljivih atributov

Problem C.20: Najmanjša $\sum$-Prilagodljivost p-Atributov

Naloga: Četvorka (G, I, C, p), kjer je $G = (V, E)$ graf prehodov scenarijev. Vozlišču $i \in V$ pripada scenarij $s_i \in I$, ki je opisan z atributi A_i. Za vsak par atributov $a_{ik} \in A_i$ in $a_{jl} \in A_j$, kjer $(i, j) \in E$, je $c_{ij}(a_{ik}, a_{jl}) \in \mathbb{Z}^+$ cena prilagajanja atributov. Parameter $p \in \mathbb{Z}^+$, da za vsak i velja $p \leq |A_i|$.
Dopustna rešitev: Zaporedje $(S_1, S_2, \ldots, S_n)$ množic $S_i \subseteq A_i$, kjer $|S_i| = p$ za vsak $1 \leq i \leq n$.
Kakovost dopustne rešitve: $f_{sum}(S_1, S_2, \ldots, S_n) = \sum_{(i,j)\in E} f_{ij}(S_i, S_j)$.
Cilj: Minimizacija.

Problem C.21: Najmanjša max-Prilagodljivost p-Atributov

Naloga: Četvorka (G, I, C, p), kjer je $G = (V, E)$ graf prehodov scenarijev. Vozlišču $i \in V$ pripada scenarij $s_i \in I$, ki je opisan z atributi A_i. Za vsak par atributov $a_{ik} \in A_i$ in $a_{jl} \in A_j$, kjer $(i, j) \in E$, je $c_{ij}(a_{ik}, a_{jl}) \in \mathbb{Z}^+$ cena prilagajanja atributov. Parameter $p \in \mathbb{Z}^+$, da za vsak i velja $p \leq |A_i|$.
Dopustna rešitev: Zaporedje $(S_1, S_2, \ldots, S_n)$ množic $S_i \subseteq A_i$, kjer $|S_i| = p$, za vsak $1 \leq i \leq n$.
Kakovost dopustne rešitve: $f_{max}(S_1, S_2, \ldots, S_n) = \max_{(i,j)\in E} f_{ij}(S_i, S_j)$.
Cilj: Minimizacija.

Problem C.22: Najmanjša $\sum$-Prilagodljivost 1-Atributov

Naloga: Trojica (G, I, C), kjer je $G = (V, E)$ graf prehodov scenarijev. Vozlišču $i \in V$ pripada scenarij $s_i \in I$, ki je opisan z atributi A_i. Za vsak par atributov $a_{ik} \in A_i$ in $a_{jl} \in A_j$, kjer $(i, j) \in E$, je $c_{ij}(a_{ik}, a_{jl}) \in \mathbb{Z}^+$ cena prilagajanja atributov.
Dopustna rešitev: Zaporedje $(a_1, a_2, \ldots, a_n)$ atributov $a_i \in A_i$.
Kakovost dopustne rešitve: $f_{sum}(a_1, a_2, \ldots, a_n) = \sum_{(i,j)\in E} c_{ij}(a_i, a_j)$.
Cilj: Minimizacija.

Problem C.23: Najmanjša max-Prilagodljivost 1-Atributov

Naloga: Trojica (G, I, C), kjer je $G = (V, E)$ graf prehodov scenarijev. Vozlišču $i \in V$ pripada scenarij $s_i \in I$, ki je opisan z atributi A_i. Za vsak par atributov $a_{ik} \in A_i$ in $a_{jl} \in A_j$, kjer $(i, j) \in E$, je $c_{ij}(a_{ik}, a_{jl}) \in \mathbb{Z}^+$ cena prilagajanja atributov.
Dopustna rešitev: Zaporedje $(a_1, a_2, \ldots, a_n)$ atributov $a_i \in A_i$.
Kakovost dopustne rešitve: $f_{max}(a_1, a_2, \ldots, a_n) = \max_{(i,j)\in E} c_{ij}(a_i, a_j)$.
Cilj: Minimizacija.

C.4 Prilagodljivostni optimizacijski problemi

Problem C.24: Najmanjši $\sum$-Prilagodljivi 1-Center

Naloga: Trojica (G, I, C), kjer je scenarij $s_i \in I$ opisan s povezanim omrežjem $G_i = (V_i, E_i)$.
Dopustna rešitev: Zaporedje $(c_1, c_2, \ldots, c_n)$, kjer $c_i \in V_i$ in velja $z_i(c_i) = z_i^*$.
Kakovost dopustne rešitve: $f_{sum}(c_1, c_2, \ldots, c_n)$.
Cilj: Minimizacija.

Problem C.25: Najmanjši max-Prilagodljivi 1-Center

Naloga: Trojica (G, I, C), kjer je scenarij $s_i \in I$ opisan s povezanim omrežjem $G_i = (V_i, E_i)$.
Dopustna rešitev: Zaporedje $(c_1, c_2, \ldots, c_n)$, kjer $c_i \in V_i$ in velja $z_i(c_i) = z_i^*$.
Kakovost dopustne rešitve: $f_{max}(c_1, c_2, \ldots, c_n)$.
Cilj: Minimizacija.

Problem C.26: Najmanjše $\sum$-Prilagodljivo Vpeto Drevo

Naloga: Trojica (G, I, C), kjer je scenarij $s_i \in I$ opisan s povezanim omrežjem $G_i = (V_i, E_i)$. Za vsak $i, j \in V$ velja $|V_i| = |V_j|$.
Dopustna rešitev: Zaporedje $(T_1, T_2, \ldots, T_n)$, kjer je $T_i \subseteq G_i$ najmanjše vpeto drevo.
Kakovost dopustne rešitve: $f_{sum}(T_1, T_2, \ldots, T_n)$.
Cilj: Minimizacija.

Problem C.27: Najmanjše max-Prilagodljivo Vpeto Drevo

Naloga: Trojica (G, I, C), kjer je scenarij $s_i \in I$ opisan s povezanim omrežjem $G_i = (V_i, E_i)$. Za vsak $i, j \in V$ velja $|V_i| = |V_j|$.
Dopustna rešitev: Zaporedje $(T_1, T_2, \ldots, T_n)$, kjer je $T_i \subseteq G_i$ najmanjše vpeto drevo.
Kakovost dopustne rešitve: $f_{max}(T_1, T_2, \ldots, T_n)$.
Cilj: Minimizacija.

Problem C.28: Najmanjši $\sum$-Prilagodljivi p-Center

Naloga: Trojica (G, I, C), kjer je scenarij $s_i \in I$ opisan z omrežjem $G_i = (V_i, E_i)$, in parameter $p \leq \min_{1 \leq i \leq n} |V_i|$.
Dopustna rešitev: Zaporedje $(C_1, C_2, \ldots, C_n)$, kjer je $C_i \subseteq V_i$ in $|C_i| \leq p$. Velja $z_i(C_i) \leq 2z_i^*$.
Kakovost dopustne rešitve: $f_{sum}(C_1, C_2, \ldots, C_n)$.
Cilj: Minimizacija.

Problem C.29: NAJMANJŠI max-PRILAGODLJIVI p-CENTER

Naloga: Trojica (G, I, C), kjer je scenarij $s_i \in I$ opisan z omrežjem $G_i = (V_i, E_i)$, in parameter $p \leq \min_{1 \leq i \leq n} |V_i|$.
Dopustna rešitev: Zaporedje $(C_1, C_2, \ldots, C_n)$, kjer je $C_i \subseteq V_i$ in $|C_i| \leq p$. Velja $z_i(C_i) \leq 2z_i^*$.
Kakovost dopustne rešitve: $f_{max}(C_1, C_2, \ldots, C_n)$.
Cilj: Minimizacija.

Problem C.30: NAJMANJŠI VEČKRITERIJSKI PRILAGODLJIVI p-CENTER

Naloga: Trojica (G, I, C), kjer je scenarij $s_i \in I$ opisan z omrežjem $G_i = (V_i, E_i)$, in parameter $p \leq \min_{1 \leq i \leq n} |V_i|$.
Dopustna rešitev: Zaporedje $(C_1, C_2, \ldots, C_n)$, kjer je $C_i \subseteq V_i$ in $|C_i| = p$.
Kakovost dopustne rešitve: $F(C_1, C_2, \ldots, C_n)$.
Cilj: Minimizacija.

Literatura

[AB97] Igor Averbakh and Oded Berman. Minimax regret p-center location on a network with demand uncertainty. *Location Science*, 5:247–254, 1997.

[AB00] Igor Averbakh and Oded Berman. Algorithms for robust 1-center problem on a tree. *European Journal of Operational Research*, 123:292–302, 2000.

[ABE05] Christian Artigues, Jean-Charles Billaut, and Carl Esswein. Maximization of solution flexiblity for robust shop scheduling. *European Journal of Operational Research*, 165:314–328, 2005.

[ABG+98] M. Ahlin, L. Bokal, A. Gložančev, M. Hajnšek-Holz, M. Humar, J. Keber, P. Kostanjevec, B. Košmrlj-Levačič, B. Lazar, J. Müller, F. Novak, Z. Praznik, J. Snoj, I. Šircelj Žnidaršič, C. Tavzes, N. Vojanovič, J. Jenčič, T. Korošec, Z. Leder, V. Majdič, J. Meze, M. Silvester, A. Vidovič-Muha, I. Kozlevčar, and P. Jakopin. *Slovar slovenskega knjižnega jezika*. Državna založba Slovenije, Ljubljana, 1998.

[ACG+99] G. Ausiello, P. Crescenzi, G. Gambosi, V. Kann, A. Marchetti-Spaccamela, and M. Protasi. *Complexity and Approximation: Combinatorial Optimization Problems and their Approximability Properties*. Springer Verlag, 1999.

[AHU74] Alfred V. Aho, John E. Hopcroft, and Jeffrey D. Ullman. *The Design and Analysis of Computer Algorithms*. Addison-Wesley, Reading, MA, 1974.

[Bat98] Vladimir Batagelj. *Diskretne strukture: zapiski predavanj*. samozaložba, Ljubljana, 1998.

[Bea85] * J. E. Beasley. A note on solving large p-median problems. *European Journal of Operational Research*, 21:270–273, 1985.

[Bel95] Erika Belehar. *Središča v grafih*. Fakulteta za matematiko in fiziko, Univerza v Ljubljani, 1995. (Diplomska naloga).

[BEY98] Allan Borodin and Ran El-Yaniv. *Online Computation and Competitive Analysis*. Cambridge University Press, Cambridge, 1998.

[BFJ94] Kathryn L. Baker, Alexander M. Franz, and Pamela W. Jordan. Coping with ambiguity in knowledge-based natural language analysis. In *Proceedings of FLAIRS-94*, 1994.

[BGKS98] Randeep Bhatia, Sudipto Guha, Samir Khuller, and Yoram J. Sussmann. Facility location with dynamic distance functions. *Journal of Combinatorial Optimization*, 2:199–217, 1998.

[BKP93] J. Bar-Ilan, G. Kortsarz, and D. Peleg. How to allocate network centers. *Journal of Algorithms*, 15:385–415, 1993.

[Cap99] Paola Cappanera. A survey on obnoxious facility location problems. Technical Report TR-99-11, 1999.
http://citeseer.nj.nec.com/cappanera99survey.html.

[CC03] Kevin M. Curtin and Richard L. Church. A family of models for multiple-type discrete dispersion. Technical Report 34/03, School of Social Sciences, The University of Texas, 2003.
http://www.utdallas.edu/dept/socsci/working_papers/wp34-03.pdf.

[Čib02] Uroš Čibej. *Vključevanje dinamičnosti in stohastičnosti v modele razmeščanja*. Fakulteta za računalništvo in informatiko, Univerza v Ljubljani, 2002. (Diplomska naloga).

[CLRS01] Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, and Clifford Stein. *Introduction to Algorithms*. MIT Press, 2. izdaja, 2001.

[Coo71] * Stephen B. Cook. The complexity of theorem-prooving procedures. In *Proc. 3rd Annual ACM Symposium on Theory of Computing, Shaker Hights*, pages 151–158, 1971.

[CR98] J. Cheriyan and R. Ravi. Approximation algorithms for network problems, 1998.
http://www.gsia.cmu.edu/afs/andrew/gsia/ravi/WWW/public/new-lecnotes.html.

[DA85] M.E. Dyer and A.M.Frieze. A simple heuristic for the p-centre problem. *Operations Research Letters*, 3:6:285–288, 1985.

[Das95] Mark S. Daskin. *Network and Discrete Location: Models, Algorithms and Applications*. Wiley, New York, 1995.

[Das00] Mark S. Daskin. A new approach to solving the vertex p-center problem to optimality: Algorithm and computational results. *Communications of the Operations Research Society of Japan*, 45:9:428–436, 2000.

[DH02] Zvi Drezner and Horst W. Hamacher, editors. *Facility Location: applications and theory*. Springer-Verlag, Berlin, 2002.

[DO99] Mark S. Daskin and Susan H. Owen. Partial covering p-center and partial covering set covering models, 1999. ISOLDE VIII Presentation, http://users.iems.nwu.edu/~msdaskin/Publications.htm.

[EBS05] Carl Esswein, Jean-Charles Billaut, and Vitaly A. Strusevich. Two-machine shop scheduling: Compromise between flexibility and makespan value. *European Journal of Operational Research*, 167:796–809, 2005.

[ELP01] Sourour Elloumi, Martine Labbe, and Yves Pochet. New formulation and resolution method for the p-center problem, 2001. http://www.optimization-online.org/DB_HTML/2001/10/394.html.

[ES04] H.A. Eislet and C.-L. Sandblom. *Decision Analysis, Location Models, and Scheduling Problems*. Springer, Berlin, 2004.

[GJ79] Michael. R. Garey and David. S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman and Co., San Francisco, 1979.

[GKP89] Ronald Lewis Graham, Donald Ervin Knuth, and Oren Patashnik. *Concrete Mathematics*. Addison-Wesley, Boston, 2. izdaja, 1989.

[GN96] Ioannis Giannikos and Stefan Nickel. Some personal views on the current state of the future of locational analysis, 1996. http://citeseer.nj.nec.com/giannikos96some.html.

[Gon85] * T. Gonzalez. Clustering to minimize the maximum intercluster distance. *Theoretical Computer Science.*, 38:293–306, 1985.

[Gov98] Manica Govekar. *Problemi razdelitve in njihova aproksimabilnost*. Fakulteta za računalništvo in informatiko, Univerza v Ljubljani, 1998. (Magistrska naloga).

[Hen99] David A. Hennessy. Capacity choice in a two-stage problem under uncertainty. *Economics Letter*, 65:177–182, 1999.

[HMU01] John E. Hopcroft, Rajeev Motwani, and Jeffrey D. Ullman, editors. *Introduction to Automata Theory, Languages, and Computation*. Addison-Wesley, Boston, 2nd 2001.

[HN98] Horst W. Hamacher and Stefan Nickel. Classification of location models. *Location Science*, 6:229–242, 1998.

[Hoc95] Dorit S. Hochbaum, editor. *Approximation Algorithms for NP-hard Problems*. PWS publishing company, Boston, 1995.

[HOH] S.M. Moattar Husseini, C. O'Brien, and S.T. Hosseini. A method to enhance volume flexibility in JIT production control. *International Journal of Production Economics*, in press.

[Hor96] John Horgan. *The End of Science*. Addison-Wesley, Reading, Massachusetts, 1996.

[HP98] Dorit S. Hochbaum and Anu Pathria. Locating centers in a dynamically changing network, and related problems. *Location Science*, 6:243:256, 1998.

[HS85] * Dorit S. Hochbaum and David B. Shmoys. A best possible heuristic for the k-center problem. *Mathematics of Operations Research*, 10:180–184, 1985.

[HS86] Dorit S. Hochbaum and David B. Shmoys. A unified approach to approximation algorithms for bottlneck problems. *Journal of ACM*, 33:533–550, 1986. PR.

[IP01] Taylan Ilhan and Mustafa C. Pinar. An efficient exact algorithm for the vertex p-center problem, 2001.
http://www.optimization-online.org/DB_HTML/2001/09/376.html.

[Kar72] * Richard M. Karp. *Complexity of Computer Computations*, chapter Recucibility among combinatorial problems, pages 85–103. Plenum Press, 1972.

[KH79] * O. Kariv and S.L. Hakimi. An algorithmic approach to network location problems. part 1: The p-centers. *SIAM Journal of Applied Mathematics*, 37:513–538, 1979.

[Knu99] Donald E. Knuth. *Fundamental Algorithms*, volume 1 of *The Art of Computer Programming*. Addison-Wesley, Reading, Massachusetts, 3. izdaja, 1999.

[KT06] Jon Kleinberg and Éva Tardos. *Algorithm Design*. Addison-Wesley, Boston, 2006.

[Kur98] Ray Kurzweil. *The Age of Spiritual Machines: When Computers Exceed Human Intelligence*. Oxford University Press, 1998.

[KV00] Bernhard Korte and Jens Vygen. *Combinatorial Optimization: Theory and Algorithms*. Springer, Berlin, 2000.

[KY97] P. Kouvelis and G. Yu, editors. *Robust Discrete Optimization and its Applications*. Kluwer Academic Publishers, Boston, 1997.

[Lin05] Snorre Lindset. Valuing the flexiblity of currency choice in multinational trade with stochastic exchange rates. *Journal of Multinational Financial Management*, 15:137–153, 2005.

[LLH05] Young Hae Lee, Moon Hwan Lee, and Sun Hur. Optimal design of rack structure with modular cell in AS/RS. *International Journal of Production Economics*, 98:172–178, 2005.

[LMV05] F. Javier Lloréns, Luis M. Molina, and Antonio J. Verdú. Flexbility of manufacturing systems, strategic change and performance. *International Journal of Production Economics*, 98:273–289, 2005.

[Meh84] Kurt Mehlhorn. *Data Structures and Algorithms 2: Graph Algorithms and NP-Completeness*. Springer-Verlag, Berlin, 1984.

[MF98] Pitu B. Mirchandani and Richard L. Francis, editors. *Discrete Location Theory*. John Wiley & Sons, New York, 1998.

[MG05] Deependra Moitra and Jai Ganesh. Web services and flexible business processes: towards the adaptive enterprise. *Information and Management*, 42:921–933, 2005.

[Mih01] Jurij Mihelič. *Aproksimacijski algoritmi za probleme razmeščanja*. Fakulteta za računalništvo in informatiko, Univerza v Ljubljani, 2001. (Diplomska naloga).

[Mih04a] Jurij Mihelič. *Hevristično reševanje problemov pokrivanja in razmeščanja*. Fakulteta za računalništvo in informatiko, Univerza v Ljubljani, 2004. (Magistrska naloga).

[Mih04b] Jurij Mihelič. *Znanstveno delo podiplomskih študentov v Sloveniji*, chapter Problemi razmeščanja ponudnikov, pages 392–398. Društvo mladih raziskovalcev Slovenije, Ljubljana, 2004.

[Min70] * E. Minieka. The m-center problem. *SIAM Review*, 12:138–139, 1970.

[MLH00] N. Mladenović, M. Labbé, and P. Hansen. Solving the p-center problem with tabu search and variable neighborhood search. Technical report, 2000.
http://smg.ulb.ac.be/Preprints/Labbe00_20.html.

[MLH03] N. Mladenović, M. Labbé, and P. Hansen. Solving the p-center problem with tabu search and variable neighborhood search. *Networks*, 42:1:48–64, 2003.

[MMRR04] Amine Mahjoub, Jurij Mihelič, Christophe Rapine, and Borut Robič. k-center problem with uncertainty: Flexible approach. *International Workshop on Discrete Optimization Methods in Production and Logistics, Omsk-Irkutsk, Russia*, 2004.

[MMRR05] Jurij Mihelič, Amine Mahjoub, Christophe Rapine, and Borut Robič. Flexible approach to the k-center problem with uncertainty. *Journal of Combinatorial Optimization (poslano v objavo)*, 2005.

[MR02a] Jurij Mihelič and Borut Robič. Approximation algorithms for k-center problem: an experimental evaluation. *Proc. OR 2002, Klagenfurt, Austria*, 2002.

[MR02b] Jurij Mihelič and Borut Robič. Approximation algorithms for k-center problem: an experimental evaluation. *Book of abstracts*, 2002. (abstract).

[MR02c] Jurij Mihelič and Borut Robič. k-center problemi. *Zbornik 11. Elektrotehniške in računalniške konference*, zv. B:2–3, 2002.

[MR03a] Jurij Mihelič and Borut Robič. Algoritmi za razmeščanje centrov. *Electrotechnical Review*, 70:3:162–166, 2003.

[MR03b] Jurij Mihelič and Borut Robič. A genetic algorithm for the k-center location problem. *European working group on locational analysis*, 2003.

[MR04a] Jurij Mihelič and Borut Robič. Algoritmi za problem najmanjega vozliščega pokritja. *Zbornik 13. Elektrotehniške in računalniške konference*, zv. B:115–118, 2004.

[MR04b] Jurij Mihelič and Borut Robič. Facility location and covering problems. *Proc. of the 7th International Multiconference Information Society IS 2004*, Volume D - Theoretical Computer Science:215–216, 2004.

[MR05a] Jurij Mihelič and Borut Robič. Flexible-attribute problems. *(poslano v objavo)*, 2005.

[MR05b] Jurij Mihelič and Borut Robič. Solving the k-center problem efficiently with a dominating set algorithm. *Journal of Computing and Information Technology*, 13(5):225–233, 2005.

[Pap94] Christos H. Papadimitriou. *Computational Complexity*. Addison-Wesley, 1994.

[Ple87] J. Plesnik. A heuristic for the p-center problem in graphs. *Discrete Applied Mathematics*, 17:263–268, 1987.

[PS98] Christos H. Papadimitriou and Kenneth Steiglitz. *Combinatorial Optimization: Algorithms and Complexity*. Dover Publications, Inc., Mineola, New York, 1998.

[QGE02] Yan Qu, Gregory Grefenstette, and David A. Evans. Resolving translation ambiguity using monolingual corpora. *Clairvoyance Corporation*, 2002.

[RE05] Charles S. ReVelle and H.A. Eiselt. Location analysis: A synthesis and survey. *European Journal of Operational Research*, 165:1–19, 2005.

[Ree95] Colin R. Reeves, editor. *Modern Heuristic Techniques for Combinatorial Problems*. McGraw-Hill, London, 1995.

[ReV97] Charles ReVelle. A perspective on location science. *Location Science*, 5:3–13, 1997.

[Rob02] Borut Robič. *Aproksimacijski algoritmi*. Založba FE in FRI, Ljubljana, 2002.

[RS03] R. Ravi and Amitabh Sinha. Hedging uncertainty: Approximation algorithms for stochastic optimization problems. Carnegie Mellon University, Pittsburg, 2003.

[Shm95] David B. Shmoys. Computing near-optimal solutions to combinatorial optimization problems. Technical report, Ithaca, NY 14853, 1995. `http://citeseer.nj.nec.com/shmoys95computing.html`.

[Ski98] Steven S. Skiena. *The Algorithm Design Manual*. Springer-Verlag, New York, 1998.

[SM96] Daniel Serra and Vladimir Marianov. The location of emergency services in changing network: The case of Barcelona. *Seventh International Symposium on Locational Decisions*, 1996.

[Sny04] Lawrence V. Snyder. Facility location under uncertainty: A review. Technical Report 04T-015, Department of Industrial and Systems Engineering, Lehigh University, 2004.

[SS] David B. Shmoys and Chaitanya Swamy. Stochastic optimization is (almost) as easy as deterministic optimization. In *45th Annual IEEE Symposium on Foundations of Computer Science*.

[Tam01] Arie Tamir. The k-centrum multi-facility location problem. *Discrete Applied Mathematics*, 109:293–307, 2001.

[Vaz01] V. Vazirani. *Approximation Algorithms*. Springer, 2001.

[Vil02] Boštjan Vilfan. *Osnovni algoritmi*. Založba FE in FRI, 2. izdaja, 2002.

[Wol02] Stephen Wolfram. *The New Kind of Science*. Wolfram Media, 2002.

[WW97] Robin J. Wilson and John J. Watkins. *Uvod v teorijo grafov*. Knjižnica Sigma, 1997.

*posredna referenca

Spisek slik in tabel

Spisek problemov

Spisek algoritmov

Stvarno kazalo

Izjava o avtorstvu

Izjavljam, da sem doktorsko disertacijo z naslovom "*Prilagodljivost v optimizacijskih problemih*" izdelal samostojno pod vodstvom mentorja prof. dr. Boruta Robiča. Izkazano pomoč drugih sodelavcev sem v celoti navedel v zahvali.

Jurij Mihelič