Univerza v Ljubljani
Fakulteta za računalništvo in informatiko

Tomaž Dobravec

Usmerjevalni algoritmi v omrežjih s topologijo krožnih grafov

DOKTORSKA DISERTACIJA

mentor: prof. dr. Borut Robič

Ljubljana, 2004

staršem

za gozd, mlinčke in storže,
za Veliko planino in Krk,
za Nairobi in Awko,
za pogum, vero
in družino ...

predvsem pa za vso ljubezen, s katero
sta me naučila stvari, pomembnejše od vsega,
kar se zdi pomembno

Deček se je igral na obličju Zemlje, vse dni od zore do mraka.
Toliko reči je bilo treba spoznati: toliko dreves pregledati, toliko oblakov zasledovati pri njihovi poti po nebu, toliko dišečih rož poduhati, toliko ptic spoznati, toliko čebel opazovati ... Predvsem pa je bilo toliko kamenčkov in vse je bilo treba prešteti, razvrstiti, morda tudi uporabiti ...

Štel je in štel in sonce je hodilo svojo pot, vsako jutro znova po isti krožnici od vzhoda do zahoda. Drevesa so ponujala svoje sadove, ptice so pele, rože so dišale, oblaki so se prevažali po nebu, čebele so nabirale med, deček pa je razvrščal kamenčke in jih štel ...
Slišal je zgodbo o Prometeju, bila je dobra, imeti ogenj je koristno in uporabno, vendar štetje zato ni nič lažje ... Slišal je zgodbo o Pitagoru, bila je dobra, da, bila je tudi uporabna, marsikaj si je lahko olajšal, kdor je razumel njegov pogled na svet ... Slišal je še mnogo zgodb, mnogi so že pred njim seštevali in razvrščali in urejali, vse si je vzel v korist, vse je uporabil, kar je razumel in potreboval ...

Leta so tekla, stoletja so tekla, tisočletja so tekla ... Zdavnaj že so ukrotili ogenj in ga uporabili, zdavnaj so že pregledali vse kamenčke, jih spoznali in smiselno razvrstili, zdavnaj že so se umaknili v velikanske sobane, kjer so stroji šteli namesto njih ...
Vse pa, kar so spoznali, je bilo osupljivo lepo; vse, kar so naredili, je bilo v resnici kar dobro; vse je teklo še kar smiselno in urejeno, vsekakor pa vse v končno korist.

Deček pa je bil večna podoba na obličju zemlje, deček, ki ni štel, da bi bilo prešteto, ki je štel, da bi se igral in radostil nad vsem ustvarjenim. Veliki programer se je od časa do časa sklonil nad Zemljo, pogledal in se nasmehnil: Bilo je zelo dobro.

md

Zahvala

Iskreno se zahvaljujem mentorju prof. dr. Borutu Robiču za vse spodbude, predloge in zamisli - vesel sem jih bil tudi takrat, ko je že kritično primanjkovalo časa za njihovo realizacijo. Hvala za res zgledno mentorstvo. Profesorju dr. Boštjanu Vilfanu se zahvaljujem za izkazano zaupanje, ko me je kot "neznanega matematika" vzel za mladega raziskovalca ter za vso podporo in pomoč v letih, ko je delo nastajalo. Za pomoč pri reševanju nekaterih matematičnih problemov ter za dolge in zanimive pogovore o krožnih grafih in njim sorodnih strukturah se zahvaljujem matematikoma prof. dr. Janezu Žerovniku in doc. dr. Primožu Potočniku. Upam, da bomo v prihodnje delo na tem področju skupaj nadaljevali.

Zahvaljujem se tudi staršema za njun zgled poštenega in trdega dela ter za prijetno družinsko ozračje, ki je ves čas moje mladosti ustvarjalo pogoje za razvoj vedoželnosti. Tedi, vem, da se z mano veseliš uspeha, saj problem, s katerim sem se ukvarjal, in njegovo rešitev poznaš mnogo bolje kot vsi, ki nanj gledamo z zemeljskimi očmi. Hvala ti za vse! Mami, hvala za vse spodbudne besede v dneh, ko nisem natančno vedel, kam naprej.
Urša, neštetokrat si me poslušala in mi dajala občutek, da razumeš moje grafe, algoritme in matematiko. Tako lepo je, če te kdo razume! Hvala za potrpežljivost, vso pozornost in predvsem za neskončno zaupanje. Zahvaljujem se tudi Sari in Maji, da sem lahko v njunem šotorčku ob zlaganju lego kock, kuhanju namišljenega kosila in branju pravljic pogosto polnil izpraznjene akumolatorje. V zahvali ne smem pozabiti Mareta, Heme in Ane - brez njih to delo nikakor ne bi bilo tako, kot je! Ata France, čeprav je XT, ki si mi ga kupil davnega leta 1986, danes že obupno zastarel, še vedno spominja na tvojo velikodušnost in vizionarsko napoved prihodnosti. Hvala, ker si mi omogočil prve korake.

Hvala tudi vsem drugim, ki jih v zahvali nisem izrecno omenil, čeprav so tudi oni posredno ali neposredno pripomogli k nastanku tega dela. Z vestnim pedagoškim in raziskovalnim delom se bom trudil upravičiti zaupanje, ki ste mi ga kot sorodniki, prijatelji, znanci ali kot davkoplačevalci izkazovali vsa leta mojega izobraževanja.

Povzetek

Doktorska disertacija *Usmerjevalni algoritmi v omrežjih s topologijo krožnih grafov* raziskuje topološke lastnosti krožnih grafov ter njihovo uporabo na področju izmenjave podatkov v računalniških omrežjih.

Krožen graf $G(n;\ \pm h_1, \ldots, \pm h_k)$ je neusmerjen, vozliščno simetričen graf nad n točkami s povezavami dolžine $h_i, \forall i = 1, 2, \ldots, k$. Natančneje, naj bodo $n, h_1, h_2, \ldots, h_k \in \mathbb{N}$ in naj velja $0 < h_1 < h_2 < \ldots < h_k < \lfloor \frac{n}{2} \rfloor$. Neusmerjen k-krožen graf $G(n;\ \pm h_1, \ldots, \pm h_k)$ je definiran kot graf, katerega množica točk je $V_n = \{0, 1, \ldots, n-1\}$, množica povezav pa množica neurejenih parov $E = \{(u, u \pm h_i);\ u \in V_n;\ i = 1, 2, \ldots, k\}$. Pri izbiri primerne topologije računalniškega omrežja predstavljajo krožni grafi vmesno možnost med preprosto a nezanesljivo topologijo obroča in zanesljivo, vendar drago ter tehnološko včasih težko izvedljivo topologijo polno povezanega omrežja. Zaradi lepih lastnosti, kot so na primer simetričnost, razširljivost, zanesljivost ter majhen premer in povprečna razdalja med točkami, krožne grafe pogosto uporabljajo kot povezovalno shemo paralelnih procesorskih sistemov, v lokalnih omrežjih ter v projektih (ILLIAC IV; FDDI-token, SILK in SONET obroči; Intel Paragon; Cray T3D; MPP; MICROS).

Usmerjevalni algoritem je postopek za reševanje problemov izmenjave podatkov v računalniških omrežjih. Podatke, ki so zapisani v obliki sporočil z natančno določenim ciljnim vozliščem, lahko razdelimo v več paketov; naloga usmerjevalnega algoritma v tem primeru je, da vse pakete po povezavah omrežja pripelje v končno vozlišče in jih tam sestavi v prvotno sporočilo. Ločimo dva tipa usmerjevalnih algoritmov: statični usmerjevalni algoritmi pred začetkom usmerjanja (to je v fazi predprocesiranja) določijo poti, po katerih bodo paketi potovali, dinamični usmerjevalni algoritmi pa te poti računajo sproti. Poleg faze predprocesiranja je usmerjevalni algoritem sestavljen tudi iz zaporedja usmerjevalnih korakov; v posameznem usmerjevalnem koraku lahko algoritem premakne pakete iz vozlišča, kjer trenutno so, v enega od sosednjih vozlišč. Pri tem mora upoštevati usmerjevalni model (ta opisuje fizične omejitve omrežja) ter morebitno zasedenost povezave (po vsaki povezavi lahko v nekem usmerjevalnem koraku potuje le en paket). Usmerjevalni algoritmi rešujejo različne usmerjevalne probleme. Najpreprostejši je problem osnovnega usmerjanja, pri katerem je v omrežju en sam paket, naloga usmerjevalnega algoritma pa je, da ta paket pripelje v njegov cilj po najkrajši poti. Mnogo težji od osnovnega usmerjanja je splošni usmerjevalni problem, pri katerem je na začetku v omrežju eden ali več paketov z natančno določenimi cilji, v katere jih mora usmerjevalni algoritem pripeljati v čim manjšem številu usmerjevalnih korakov.

Uvodnemu poglavju doktorske disertacije sledi poglavje definicij, ki je namenjeno predstavitvi širšega področja dela. V njem so podrobneje opisani pojmi omrežje, usmerjevalni model, usmerjevalni problem ter usmerjevalni algoritem, podana pa je tudi mera (t.j. cenilna funkcija), ki smo jo v delu uporabljali za ocenjevanje kakovosti posameznega usmerjevalnega algoritma.

V poglavju o krožnih grafih se natančneje posvetimo izbrani topologiji, opišemo znane rezultate, najpomembnejše topološke lastnosti in področja uporabe. Nato predstavimo

način reševanja problemov premera in najkrajše poti med danim parom točk v dvokrožnih grafih s pomočjo celoštevilske mreže, v kateri posamezna točka predstavlja sprehod v grafu. Uvedemo tudi pojem minimalne projekcije, ki se nanaša na iskanje predstavnikov celoštevilske mreže z določenimi lastnostmi, ter predstavimo algoritem za njen izračun. Celoštevilsko mrežo predstavimo kot dvodimenzionalni modul, definiramo pojem pakirane baze podmodula krožnih sprehodov in predstavimo dva algoritma za njen izračun. Nato opišemo in dokažemo nekatere pomembne lastnosti pakirane baze, povezane z medsebojno lego baznih vektorjev in s karakterističnimi koti. V nadaljevanju definiramo pojem delno usmerjenih krožnih grafov in omejenih najkrajših poti ter predstavimo $\mathcal{O}(\log n)$ algoritem za izračun teh poti v dvokrožnih grafih. Dokaz pravilnosti omenjenega algoritma, ki rešitev poišče z večkratno uporabo minimalne projekcije vzdolž vektorjev pakirane baze, temelji na lastnostih pakirane baze, ki so predstavljene v prejšnjih razdelkih. V poglavju opišemo tudi, kako nekatere pojme, ki se nanašajo na dvokrožne grafe (prevedba problema, minimalna projekcija, iskanje predstavnika množice X_w), posplošimo na splošne k-krožne grafe.

V poglavju o usmerjanju podatkov v krožnih grafih najprej opišemo znan statični in nov dinamični algoritem za osnovno usmerjanje. V nadaljevanju podamo model usmerjevalnih algoritmov ter pet statičnih algoritmov za reševanje splošnega usmerjevalnega problema v dvokrožnih grafih. Algoritme zapišemo v modelu in podamo nabor parametrov za vsakega od njih. Opišemo tudi dinamični usmerjevalni algoritem in ga primerjamo z najboljšim statičnim. Na koncu poglavja podamo posplošitev dinamičnega algoritma za uporabo v splošnih k-krožnih grafih, predstavimo algoritem za izračun približkov omejenih najkrajših poti v teh grafih ter opišemo rezultate simulacij.

Zadnje poglavje je namenjeno predstavitvi nekaterih odprtih vprašanj in predlogov za nadaljevanje dela ter predstavitvi avtorjevih izvirnih prispevkov k znanosti.

Abstract

The thesis *Routing algorithms in networks with the topology of circulant graphs* investigates topological properties of circulant graphs and their application to computer data exchange.

Circulant graphs are undirected vertex transitive graphs with n vertices and edges of length $h_i, i = 1, 2, \ldots, k$. More precisely, let for $n, h_1, h_2, \ldots, h_k \in \mathbb{N}$ be $0 < h_1 < h_2 < \ldots < h_k < \lfloor \frac{n}{2} \rfloor$. Then, the undirected k-circulant graph $G(n; \pm h_1, \ldots, \pm h_k)$ has vertex set $V_n = \{0, 1, \ldots, n-1\}$ and edge set which is the union of sets of unordered pairs $E_i = \{(u, u + \text{ sign}(i) * h_{|i|} \pmod n) | u \in \mathbb{Z}_n\}$, for $i \in \{-k, \ldots, -1, 1, \ldots, k\}$. When selecting an appropriate topology for a computer network, circulant graphs represent an intermediate choice between simple but unreliable ring topology and reliable but expensive (and sometimes technologically unfeasible) fully connected topology. Due to their favorable properties–among which are symmetry, scalability, reliability, small diameter, and small average node distance,–circulant graphs are widely studied as suitable topologies for local area networks and parallel computer architectures. They have been efficiently used in ILLIAC IV, FDDI-token, SILK and SONET rings, Intel Paragon, Cray T3D, MPP, and MICROS.

Routing algorithms are used to solve data exchange problems in computer networks. Data (i.e. a message) is split into several packages. The task of a routing algorithm is to transfer the packages along the network edges to their final destination and then to combine them into original message. There are two types of routing algorithms. Static routing algorithms determine the path for each packet in the preprocessing phase (i.e. before the actual start of routing); dynamic routing algorithms perform all the necccessary routing calculations on-the-fly. After the preprocessing phase the routing algorithm makes several successive routing steps. In each routing step the algorithm can move a packet from its current node to one of the neighboring nodes. Routing algorithms can be used to solve various *routing problems*. A *two-terminal* routing is a problem where only one package has to be routed from its source to its final destination. Routing algorithms for this problem aim to route a package along one of its shortest paths. Much more complicated situation occurs when more than one package is present in the network. In this case, which is also known as the *general routing problem*, a routing algorithm tries to route all the packages to their final destination in as few as possible routing steps.

In this thesis we first introduce the wide area of our work. We extensively explain the meaning of the following terms: network, routing model, routing problem, routing algorithm. We also present the criteria for measuring the quality of a given routing algorithm. In the following there are two main chapters. The first one is dedicated to the selected topology (i.e. circulant graphs) and the second one to the routing algorithms designed and tested for this topology.

In the chapter dedicated to circulant graphs we first summarize the known results and the most important topological properties. Then we present a method for solving certain problems (such as calculating the diameter and calculating the shortest paths) by reduction to integer lattice in which a point represents a walk in a graph. We introduce the notion of

minimal projection and give the corresponding algorithm to construct it. We present the integer lattice as a module, define the notion of packed basis, and present two algorithms for calculating it. Then we introduce and prove several important features concerning vectors of packed basis. We also introduce a new topology, the so called semi-directed circulant graphs, and present an $\mathcal{O}(\log n)$ algorithm for calculating the shortest paths between nodes in these graphs. The proof of correctness for this algorithm, which calculates the shortest paths by using the minimal projection along the vectors of packed basis, is based on the properties of packed basis proved in earlier sections. We also present some k-generalizations of the algorithms presented for 2-circulant graphs.
In the chapter on the routing in circulant graphs we first present two routing algorithms for the two-terminal routing problem (static and dynamic version). Then we present a parameterized model used to describe routing algorithms and give five parameter-sets of five static routing algorithms for the general routing problem. We also describe a dynamic algorithm for the general routing problem and compare the efficiency of this algorithm with the best static one. At the end of the chapter we present a generalization of a routing algorithm for general k-circulant graphs.
In the last chapter of the thesis we give some open questions and ideas for future work.

Kazalo

Povzetek **i**

Abstract **iii**

1 Uvod **1**
1.1 Predstavitev dela . . . 1
1.2 Dogovorjene oznake . . . 3

2 Omrežja in usmerjanje podatkov **5**
2.1 Omrežja . . . 6
2.1.1 Vozlišča in povezave omrežja . . . 7
2.1.2 Osnovne definicije . . . 8
2.1.3 Predstavitev nekaterih omrežij . . . 8
2.2 Usmerjevalni modeli . . . 10
2.2.1 Paketno usmerjanje . . . 11
2.2.2 Tip povezave . . . 12
2.2.3 Stopnja vzporednosti . . . 12
2.2.4 Reševanje trkov . . . 12
2.3 Usmerjevalni problemi . . . 13
2.3.1 Splošni usmerjevalni problem . . . 13
2.4 Usmerjevalni algoritmi . . . 15
2.5 Ocenjevanje kakovosti algoritmov . . . 16

3 Krožni grafi **19**
3.1 Kaj je krožen graf? . . . 19
3.2 Definicije in dokaz njihove ekvivalentnosti . . . 22
3.3 Uporaba krožnih grafov in primeri . . . 27
3.4 Lastnosti krožnih grafov . . . 28
3.4.1 Predstavitev . . . 28
3.4.2 Simetrija . . . 29
3.4.3 Povezanost . . . 30
3.4.4 Premer . . . 31
3.5 Sprehodi in poti v krožnih grafih . . . 32

3.5.1 Vrstni red povezav vzdolž sprehoda . . . 32
3.5.2 Regularni sprehodi . . . 33
3.5.3 Ekvivalentni sprehodi . . . 34
3.5.4 Iskanje sprehoda in kongruenčna enačba . . . 35
3.5.5 Pot in najkrajša pot v krožnih grafih . . . 35
3.6 Dvokrožni grafi in mreža $\mathbb{Z}_2^2$ . . . 36
3.6.1 Označevalna funkcija . . . 37
3.6.2 $\mathbb{Z}_2^2$ kot modul . . . 38
3.6.3 Baza modula X . . . 39
3.6.4 Računanje baze modula X . . . 41
3.7 Predstavitev množice X_w . . . 43
3.7.1 Ciklična predstavitev . . . 43
3.7.2 Vzporedna predstavitev . . . 43
3.8 Iskanje predstavnika množice X_w . . . 47
3.8.1 Rešitev kongruenčne enačbe . . . 47
3.8.2 Iskanje osnovnega baznega elementa . . . 49
3.9 Projekcije v celoštevilski mreži . . . 50
3.9.1 Minimalna projekcija v $\mathbb{Z}_2^2$ . . . 50
3.9.2 Najmanjši element na traku v $\mathbb{Z}_2^2$ . . . 52
3.9.3 Minimalna projekcija v k-dimenzijah . . . 54
3.10 Pakirana baza . . . 55
3.10.1 Eksplicitna formula za izračun pakirane baze . . . 55
3.10.2 Tipizacija pakiranih baz . . . 59
3.10.3 Lastnosti pakirane baze . . . 60
3.11 Premer in najkrajše poti v dvokrožnih grafih . . . 65
3.11.1 Dolžina najkrajše poti . . . 65
3.11.2 Premer . . . 68
3.11.3 Najkrajša pot . . . 68
3.12 Delno usmerjeni krožni grafi in omejene poti . . . 69
3.13 Omejene najkrajše poti v dvokrožnih grafih . . . 73
3.13.1 Na poti do rešitve . . . 73
3.13.2 Rešitev problema . . . 76
3.13.3 Strategija dokazovanja . . . 77
3.13.4 Dokaz pravilnosti . . . 79

4 Paketno usmerjanje v krožnih grafih 97
4.1 Osnovno usmerjanje . . . 100
4.1.1 Statičen algoritem . . . 101
4.1.2 Dinamičen algoritem . . . 102
4.2 Splošni usmerjevalni problem . . . 105
4.2.1 Model usmerjevalnih algoritmov . . . 106
4.2.2 Statični usmerjevalni algoritmi $\mathcal{S}_0$, $\mathcal{S}_1$, $\mathcal{S}_2$, $\mathcal{S}_3$ in $\mathcal{S}_4$ . . . 112
4.2.3 Dinamični usmerjevalni algoritem $\mathcal{D}$ . . . 119

4.2.4 Kakovost usmerjevalnih algoritmov v dvokrožnih grafih 120
4.3 Dinamično usmerjanje v k-krožnih grafih 122
4.3.1 Omejene najkrajše poti v k-krožnih grafih 123
4.3.2 Kakovost usmerjevalnih algoritmov v k-krožnih grafih 127

5 Zaključek **131**
5.1 Prispevki k znanosti 131
5.2 Nadaljevanje dela 133

Izjava o avtorstvu **135**

Literatura **136**

Poglavje 1

Uvod

Krožni grafi so zaradi lepih lastnosti predmet mnogih teoretičnih raziskav na matematičnem (teorija grafov, algebra) in tudi na računalniškem področju (uporaba krožnih grafov kot topologije za računalniška omrežja ter ugotavljanje sposobnosti učinkovite izmenjave podatkov med vozlišči teh omrežij). Namen doktorske disertacije *Usmerjevalni algoritmi v omrežjih s topologijo krožnih grafov* je dopolniti te raziskave z novimi odkritji, predvsem na področju, ki je v literaturi manj opisano, to je področje usmerjanja v *neusmerjenih* krožnih grafih. Ker brez matematičnega formalizma do nekaterih rezultatov ni mogoče priti oziroma se jih ne da ustrezno utemeljiti, je del te disertacije močno matematično obarvan. Definicije, trditve in dokaze, povezane z lastnostmi krožnih grafov in sorodnih struktur, uporabimo pri razvoju in dokazu pravilnosti učinkovitega usmerjanja v dvokrožnih in splošnih k-krožnih grafih.

1.1 Predstavitev dela

Delo sestavljajo tri večja poglavja: poglavje o omrežjih in usmerjanju podatkov, v katerem predstavimo širše področje disertacije; poglavje o krožnih grafih, v katerem se osredotočimo na izbrano topologijo in predstavimo mnoge njene lastnosti; ter poglavje o usmerjanju v krožnih grafih, kjer opišemo nekatere usmerjevalne algoritme za osnovno in splošno usmerjanje ter primerjamo njihovo učinkovitost.
Prvo od omenjenih poglavij je zelo splošno in je namenjeno spoznavanju osnovnih pojmov s širšega področja dela. Bralec, ki mu usmerjanje podatkov v računalniških omrežjih ni tuje, lahko to poglavje brez škode preskoči, če pa pojmov v naslednjih poglavjih ne bi razumel, naj v njem poišče razlago.
V poglavju o krožnih grafih najprej predstavimo štiri pogosto uporabljene definicije (KGA, KGB, KGC in KGD) krožnih grafov in pokažemo, da so med seboj ekvivalentne (to je, da opisujejo isto družino grafov). Nato opišemo nekaj znanih primerov krožnih grafov ter naštejemo računalnike in računalniške sisteme, v katerih te uporabljamo. Sledijo razdelki, v katerih opišemo lastnosti krožnih grafov (simetrija, povezanost, premer), definiramo pojme sprehod, regularni sprehod, ekvivalentni sprehodi in pot ter pokažemo zvezo med iskanjem sprehoda v krožnih grafih in reševanjem linearne

kongruenčne enačbe. V razdelku 3.6 predstavimo celoštevilsko mrežo $\mathbb{Z}_2^2$ kot okolje, v katerega lahko prevedemo probleme, povezane s sprehodi v krožnih grafih (na primer računanje premera ali računanje najkrajše poti med danim parom točk). V njem omenimo tudi, da je $\mathbb{Z}_2^2$ celoštevilski modul, da je množica vseh elementov, ki predstavljajo krožne sprehode (to je množica X), njegov dvodimenzionalni podmodul ter da v X obstaja dvodimenzionalna baza. Sledi razdelek, v katerem definiramo pojem minimalne projekcije ter predstavimo algoritem za njen izračun. V razdelku 3.10 definiramo poseben tip baze za podmodul X (pakirana baza), opišemo dva algoritma za njen izračun (iterativni algoritem in eksplicitno formulo), podamo tipizacijo pakiranih baz ter predstavimo nekatere pomembne lastnosti, povezane s smerjo diagonale baznega paralelograma in s karakterističnimi koti. Pakirana baza in minimalna projekcija sta pomembni orodji v algoritmih za računanje najkrajših poti v krožnih grafih, ki jih predstavimo v razdelkih 3.11 in 3.13. V prvem pokažemo, kako se problema najkrajša pot in premer prevedeta v okolje $\mathbb{Z}_2^2$ in kako ju tam s pomočjo pakirane baze tudi rešimo, v drugem pa predstavimo algoritem za izračun omejenih najkrajših poti v dvokrožnih grafih (te so definirane v razdelku 3.12). Ta algoritem poleg prevedbe v $\mathbb{Z}_2^2$ in pakirane baze uporablja tudi minimalno projekcijo. V trditvi 3.13-I (razdelek 3.13.1) dokažemo pravilnost algoritma za izračun omejenih najkrajših poti; na njeni osnovi temelji dinamični usmerjevalni algoritem $\mathcal{D}$, ki ga predstavimo v nadaljevanju. Pri dokazovanju trditve 3.13-III (razdelek 3.13.4), iz katere neposredno sledi pravilnost trditve 3.13-I, uporabimo prijeme in orodja iz prejšnjih razdelkov tega poglavja (prevedba problema, lastnosti pakirane baze, minimalna projekcija, reševanje kongruenčne enačbe in drugo).

V poglavju o usmerjanju v krožnih grafih se najprej osredotočimo na usmerjevalne algoritme za reševanje osnovnega usmerjevalnega problema. V razdelku 4.1 predstavimo statičen algoritem $\mathcal{SB}$, ki paket usmerja po eni izmed poti ekvivalenčnega razreda, določenega v fazi predprocesiranja, in dinamičen algoritem $\mathcal{RSP}$, ki pot paketa računa sproti (pri tem uporablja algoritem za izračun omejenih najkrajših poti). Pravilnost delovanja algoritma $\mathcal{RSP}$ je dokazana v razdelku 4.1.2. S splošnim usmerjevalnim problemom se ukvarjamo v razdelku 4.2, kjer najprej definiramo enoten parametriziran model za zapis usmerjevalnih algoritmov (razdelek 4.2.1). Algoritmi, zapisani v tem modelu, so med seboj dobro primerljivi, poleg tega je implementacija takih algoritmov v simulatorju zelo preprosta. V nadaljevanju (razdelek 4.2.2) predstavimo pet statičnih usmerjevalnih algoritmov ($\mathcal{S}_0$, $\mathcal{S}_1$, $\mathcal{S}_2$, $\mathcal{S}_3$ in $\mathcal{S}_4$), ki vsi uporabljajo isto metodo za določitev nadaljevanja poti posameznega paketa (NASLEDNJA_POVEZAVA$_{\mathcal{SB}}$), razlikujejo pa se v nastavitvah drugih parametrov modela. V razdelku 4.2.3 predstavimo algoritem $\mathcal{D}$, ki uporablja iste parametre modela kot (glede na naše simulacije) najboljši statični algoritem $\mathcal{S}_4$, nadaljevanje poti posameznega paketa pa določi dinamično s pomočjo metode NASLEDNJA_POVEZAVA$_{\mathcal{RSP}}$. Ugotovitve o kakovosti usmerjevalnih algoritmov, ki smo jih dobili s pomočjo simulatorja, so predstavljene v razdelku 4.2.4. V zadnjem razdelku tega poglavja opišemo posplošitev algoritma $\mathcal{D}$ za uporabo v splošnih k-krožnih grafih. Predstavimo tudi algoritem za izračun približkov omejenih najkrajših poti v k-krožnih grafih in primerjamo kakovost približkov z natančnimi vrednostmi. Na koncu poglavja predstavimo rezultate simulacij dinamičnega usmerjanja v k-krožnih grafih.

1.2 Dogovorjene oznake

V delu poleg splošno uveljavljenih uporabljamo tudi naslednje matematične oznake:

$$\text{sign}(x) = \begin{cases} 1; & x > 0 \\ -1; & x < 0 \\ 0; & x = 0 \end{cases} \qquad \text{predznak števila (\textit{angl. signum})}$$

$$\text{parity}(x) = \begin{cases} 1; & x \text{ je sod} \\ -1; & x \text{ je lih} \end{cases} \qquad \text{sodost števila (\textit{angl. parity})}$$

$$\lfloor x \rfloor = \max\{n \in \mathbf{Z};\ n \le x\} \qquad \text{zaokroževanje navzdol (\textit{angl. floor})}$$

$$\lceil x \rceil = \min\{n \in \mathbf{Z};\ n \ge x\} \qquad \text{zaokroževanje navzgor (\textit{angl. ceiling})}$$

$$[x] = \text{sign}(x)\lfloor |x| \rfloor \qquad \text{celi del (\textit{angl. integer part})}$$

$$\lfloor x \rceil = \lfloor x + 0.5 \rfloor \qquad \text{najbližje celo število (\textit{angl. round})}$$

Slike, tabele, algoritmi, primeri ter matematični pojmi (definicije, izreki, trditve, leme in posledice) so številčeni vsak s svojim števcem, ki je sestavljen iz številke razdelka (arabska številka) ter zaporedne številke objekta v trenutnem razdelku (rimska številka). Tako je na primer četrta slika v razdelku 2.3 označena z 2.3.IV. Pomemben del disertacije predstavljajo matematični dokazi, ki so označeni, kot sledi.

Dokaz. Začetek posameznega dokaza je označen z znakom '⌈', konec pa z znakom '⌊' (oziroma '⌉' in '⌋') ob zunanjem robu besedila. Včasih, kadar se dokaz razteza čez več kot eno stran, je ob znaku za začetek dokaza napisana tudi številka strani, na kateri se dokaz konča, ob znaku za konec dokaza pa številka strani, na kateri se je dokaz začel. Ta način omogoča hitrejše branje besedila s preskakovanjem dolgih in zapletenih dokazov. Namesto izraza Q.E.D. ("Quod Erat Demonstrandum" oziroma "Quonc Enga Dokaza") uporabljamo znak □. Znak —⋆—označuje protislovje. □

⌉3
⋮
⋮
⌋3

Poglavje 2

Omrežja in usmerjanje podatkov

Glavni namen računalniškega omrežja, ki povezuje dve ali več samostojnih delovnih enot v celoto, je ustvariti okolje, ki bo omogočalo izmenjavo podatkov med posameznimi enotami. Način izmenjave podatkov je pomemben, ko so v omrežje povezani posamezni računalniki (LAN, telekomunikacijska omrežja, gruče), pa tudi takrat, ko so vanj povezani procesorji večprocesorskega sistema. Optimizacija izmenjave podatkov je posebno pomembna takrat, ko delovne enote računalniškega omrežja uporabljamo za hkratno reševanje istega problema, saj je komunikacija med delovnimi enotami pogosto ozko grlo vzporednega računanja. Na način posredovanja vhodnih podatkov in kakovost izmenjave delnih rezultatov vplivajo topologija omrežja, usmerjevalni model, usmerjevalni problem, usmerjevalni algoritem in kriterij za ocenjevanje kakovosti [18, 19].

Topologija omrežja. Topologija je fizična povezanost omrežja. Čeprav današnja tehnologija že omogoča spreminjanje te lastnosti na elektronski način, bomo zaradi nazornosti tako poimenovanje ohranili. Od topologije so odvisne različne lastnosti: premer, stopnja vozlišč, povezanost, razširljivost, odpornost na napake in podobno [44, 52]. Najbolj znane topologije so mreže, torusi, hiperkocke, cikli, drevesa, krožni grafi [2, 43, 44, 52].

Usmerjevalni model. Usmerjevalni model je podan z množico pravil, ki določajo

- strategijo prenosa podatkov (preklapljanje povezav (*angl. circuit switching*), usmerjanje v omrežju z vodili (*angl. routing along a network of buses*), paketno usmerjanje (*angl. packet routing*)),
- tip povezav (usmerjene (*angl. directed*), neusmerjene (*angl. undirected*), delno usmerjene (*angl. semi-directed*), hkratno dvosmerne (*angl. full-duplex*)),
- čas prenosa podatkov (konstanten ali linearen glede na velikost podatkov),
- pomnilniško kapaciteto posameznega vozlišča (velikost čakalnih vrst),
- stopnjo vzporednosti (enosmerna (*angl. SIMD*) ali večsmerna komunikacija (*angl. MIMD*)) in podobno [34].

Usmerjevalni model je odvisen predvsem od strojne opreme in ga pri danem omrežju običajno ni mogoče spreminjati.

Usmerjevalni problem. Hitrost usmerjanja je odvisna tudi od usmerjevalnega problema, ki ga trenutno rešujemo. Tako bo, na primer, hitrost prenosa pri velikem številu podatkov slabša kot takrat, ko bo teh podatkov malo – veliko število podatkov namreč pomeni veliko zasedenost posameznih virov omrežja in zato veliko čakanja, usklajevanja, sklepanja kompromisov. Podobno bo takrat, ko bodo podatki v omrežju razporejeni enakomerno, hitrost prenosa boljša kot takrat, ko bo večina podatkov zbranih v nekaj vozliščih. Ker usmerjevalni problem med drugim določa tudi količino in razporeditev podatkov, posredno vpliva na hitrost prenosa. Usmerjevalni problemi so oddajanje (*angl. broadcasting*), zbiranje (*angl. gathering*), sporočanje (*angl. scattering*), permutacijsko usmerjanje (*angl. permutation routing*), izotonično usmerjanje (*angl. isotonic routing*), splošni usmerjevalni problem (*angl. general routing problem*) [34].

Usmerjevalni algoritem. Naloga usmerjevalnega algoritma je, da reši usmerjevalni problem, to je, da za vsak podatek določi, kako bo potoval po omrežju, da bo dosegel svoj cilj. Pri tem upošteva vse omejitve, ki jih narekujeta omrežje in usmerjevalni model, poleg tega pa upošteva prisotnost drugih podatkov ter prepreči oziroma odpravi posledice trkov. Usmerjevalne algoritme delimo na statične, ki pred dejanskim začetkom usmerjanja (t.j. v času predprocesiranja) določijo parametre potovanja za posamezen paket, in dinamične, ki pot posameznega podatka določajo sproti (t.j. na posameznem usmerjevalnem koraku). Ker je rešitev danega usmerjevalnega problema veliko, usmerjevalni algoritmi iščejo tako, ki je čim boljša v smislu danega kriterija kakovosti.

Kriterij kakovosti. V praksi uporabljamo različne kriterije za merjenje kakovosti usmerjevalnih algoritmov, na primer: največja ali povprečna dolžina poti, skupno število usmerjevalnih korakov, povprečni čakalni čas, največja dolžina čakalne vrste in podobno [13].

Opomba. Izraz *trk* se v tem delu nanaša na situacijo, ko se dva paketa potegujeta za isti vir (povezavo, mesto v čakalni vrsti in podobno). Nekateri namesto izraza trk uporabljajo izraz *kolizija*.

V nadaljevanju tega poglavja bomo omenjene dejavnike, od katerih sta odvisna način in kakovost izmenjave podatkov, podrobneje predstavili.

2.1 Omrežja

Pri povezovanju delovnih enot v omrežje z namenom, da bi ustvarili sistem za vzporedno reševanje istega problema, je izbira primerne povezovalne sheme bistvena [2, 34]. Komu-

nikacija med enotami je časovno zahtevna operacija in je ozko grlo vzporednega sistema, zato je zaželjeno, da ima sistem čim več neposrednih povezav med enotami. Vendar pa je bogata povezovalna shema draga in v praksi zaradi tehnoloških in naravnih ovir celo nemogoča (zaradi pomanjkanja prostora, problemov hlajenja, predolgih povezav ...). Najti je torej treba kompromis med omenjenima nasprotujočima si zahtevama, poleg tega pa je treba upoštevati še specifičnost problema, zaradi katerega povezujemo delovne enote.

Osnova sistema za vzporedno računanje je **omrežje**, ki je sestavljeno iz vozlišč in povezav med njimi, pri čemer so vozlišča delovne enote omrežja, povezave pa fizične povezave med vozlišči, po katerih je mogoče poslati sporočila. Omrežja modeliramo z **grafi**; točka $v \in V$ grafa $G = (V, E)$ je eno izmed vozlišč, povezava $e \in E$ pa ena izmed povezav omrežja [56]. **Topologija omrežja** je množica vseh parov vozlišč (v_i, v_j) danega omrežja, med katerimi obstaja neposredna fizična povezava [53]. Čeprav omenjeni trije pojmi (omrežje, graf, topologija) niso sinonimi, jih v literaturi izmenično uporabljajo za opisovanje iste strukture: množice delovnih enot, povezanih v en sistem z možnostjo izmenjave podatkov med enotami.

2.1.1 Vozlišča in povezave omrežja

Vozlišča v omrežju

Osnovni gradniki omrežja so njegova vozlišča. Vsako vozlišče je ena izmed samostojnih delovnih enot, na primer eden od procesorjev večprocesorskega sistema ali pa eden izmed računalnikov, ki so povezani v omrežje. Vozlišče je sposobno samostojno izvajati algoritme. Vozlišča so običajno opremljena z lokalnim pomnilnikom, ki ga uporabljajo, na primer, za shranjevanje lokalnih parametrov, rezultatov izvajanja algoritmov, implementacijo čakalnih vrst in podobno.

Kadar omrežja obravnavamo s teoretičnega stališča, *vozlišča* pogosto imenujemo kar *točke*, pri tem pa imamo v mislih točke grafa, s katerim dano omrežje modeliramo.

Povezave med vozlišči

Poznamo dve vrsti povezav med vozlišči omrežja. **Dvotočkovna** povezava povezuje dve vozlišči omrežja (oddajnik in sprejemnik), **skupinska** povezava pa omogoča priključitev več vozlišč na isti prenosni medij (vodilo). Pri prvi vrsti povezav vozlišče (oddajnik) pošlje sporočilo, ki je namenjeno natanko enemu vozlišču (tistemu, ki je na drugi strani povezave – sprejemniku), pri drugi vrsti pa vozlišče (oddajnik) pošlje sporočilo, ki je namenjeno enemu ali več vozliščem, ki so priklopljena na isto vodilo; vozlišča ves čas "prisluškujejo" vodilu in čakajo, da se bo na njem pojavilo sporočilo, ki je namenjeno (tudi) njim.

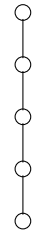
Dvotočkovna povezava med vozliščema lahko služi kot **enosmerna** ali kot dvosmerna povezava. Prva povezava, ki jo imenujemo tudi **usmerjena**, služi za prenos podatkov v eni smeri, po drugi, **neusmerjeni** povezavi, pa lahko podatke prenašamo v obeh smereh, toda ne hkrati. Povezavo, po kateri lahko podatke prenašamo v obeh smereh hkrati, imenujemo **hkratna dvosmerna** povezava.

2.1.2 Osnovne definicije

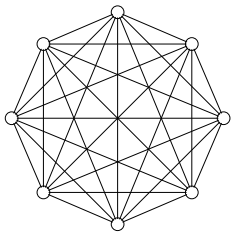
Za dve vozlišči omrežja pravimo, da sta **sosednji**, če je med njima neposredna povezava. **Stopnja** vozlišča je število njegovih sosed. Od omrežja običajno zahtevamo, da je **povezano**, kar pomeni, da je med poljubnima dvema vozliščema pot. **Razdalja** med dvema vozliščema je dolžina najkrajše poti med njima, **premer** (*angl. diameter*) omrežja pa je največja razdalja med kakim parom vozlišč. Od "dobrega" omrežja pričakujemo, da bo imelo čim manjši premer pri kar se da majhnem številu povezav, hkrati pa, da bo čim bolj **razširljivo** (*angl. scalable*) in čim manj **občutljivo na napake** (*angl. fault tolerant*) [4]. Dobra razširljivost omrežja pomeni, da ga lahko razširimo z dodajanjem novih vozlišč, ne da bi s tem bistveno spremenili premer (kot funkcijo števila vozlišč) in stopnje vozlišč [43], neobčutljivost na napake pa pomeni, da se premer ne spremeni bistveno in da omrežje ostane povezano tudi takrat, ko ena ali več povezav zaradi napake izpade [2]. Že dejstvo, da z redčenjem povezav premer omrežja narašča, daje slutiti, da "idealnega" omrežja ni: bodisi ima omrežje veliko povezav in je zato slabo razširljivo, ima majhen premer in je neobčutljivo na napake bodisi ima malo povezav in je dobro razširljivo, toda bolj občutljivo na napake in ima velik premer.

2.1.3 Predstavitev nekaterih omrežij

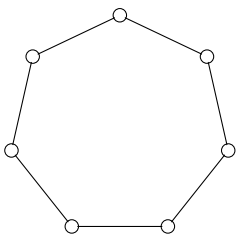
Veriga je omrežje, v katerem je vozlišče 0 povezano z vozliščem 1, vozlišče 1 z vozliščem 2, ..., vozlišče $n-2$ pa z vozliščem $n-1$. Veriga ima torej $n-1$ povezav, prav tak je njen premer. Izpad katerekoli povezave razdeli omrežje v dve nepovezani množici, dodajanje novih vozlišč pa je zelo preprosto.



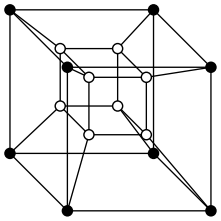
Povsem nasprotne značilnosti ima **polno povezano omrežje**, v katerem je vsako vozlišče povezano z vsemi drugimi. Število povezav je $n(n-1)/2$, premer je 1, ne glede na število vozlišč. V praksi verigo in polno povezano omrežje redko uporabljamo, običajno izberemo neko vmesno možnost.



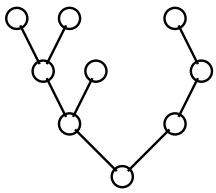
Cikel (obroč) je dopolnjena veriga – dodatno povežemo še prvo in zadnje vozlišče. Premer se razpolovi, izpad ene povezave pa ni več usoden.



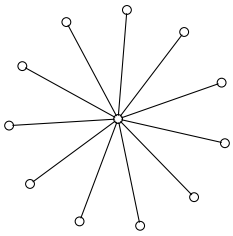
Zaradi dobre razširljivosti in sorazmerno majhnega premera se pogosto uporablja omrežje **hiperkocke** reda k oziroma k-dimenzionalna hiperkocka, z $n = 2^k$ vozlišči in premerom ter stopnjo vsakega vozlišča k. 1-hiperkocka je veriga z dvema vozliščema, za $k > 1$ pa k-hiperkocko dobimo tako, da vzamemo dve $(k-1)$-hiperkocki in povežemo soležna vozlišča. Strukturo hiperkocke, ki smo jo podali z rekurzivno (geometrično) definicijo, lahko opišemo tudi drugače. Če namreč vsako od 2^k vozlišč k-dimenzionalne hiperkocke enolično predstavimo s k-bitnim binarnim zapisom, potem velja, da so sosednja vozlišča tista, ki se med seboj razlikujejo natanko v enem bitu binarne predstavitve.



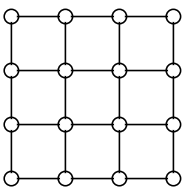
Drevo je povezano omrežje brez ciklov, oziroma omrežje, v katerem med poljubnima dvema vozliščema obstaja natanko ena pot. Tako omrežje je zelo občutljivo na napake. Drevo ima dve vrsti vozlišč: notranja vozlišča in liste. Vozlišče, ki ni list, ima lahko poljubno stopnjo, listi imajo vedno stopnjo 1. V posebnem primeru, ko imajo vsa notranja vozlišča stopnjo 2 ali 3, govorimo o **binarnem drevesu**. Nesimetrija drevesa se odraža tudi v tem, da imajo povezave različno težo pri ohranjanju funkcionalnosti omrežja. Če namreč izpade povezava, ki povezuje neki list s preostalim omrežjem, bo omrežje veliko manj prizadeto, kot če izpade povezava, ki povezuje notranje vozlišče s katerim od njegovih sinov.



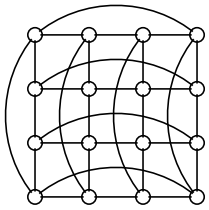
Poseben primer drevesa je **zvezda**. Od n vozlišč jih je $n-1$ povezanih s središčnim vozliščem, ki ima zato stopnjo $n-1$, preostala pa stopnjo 1; premer zvezde je 2 ne glede na število vozlišč, razširljivost je dobra. V nasprotju s povezavami, ki so v primeru zvezde vse enako pomembne za ohranjanje povezanosti omrežja, igra središče veliko pomembnejšo vlogo kot druga vozlišča. Ves promet podatkov namreč poteka prek središča, zato je izpad tega vozlišča usoden za omrežje.



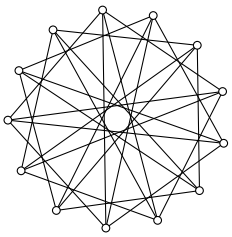
Ravninska mreža je sestavljena iz $n = p * q$ vozlišč, razporejenih v q vrsticah po p vozlišč. Vozlišča so povezana s svojimi najbližjimi sosedami; notranja vozlišča imajo tako stopnjo štiri, oglišča imajo stopnjo dva, preostala robna vozlišča pa tri.



Da bi dobili vozliščno simetrično omrežje, v mrežo dodamo dodatne povezave (krožno povezovanje). S krožnim povezovanjem vrstic mreže (vsako vozlišče prvega stolpca mreže povežemo z vozliščem v zadnjem stolpcu iste vrstice), dobimo omrežje, imenovano **valj**. Če v valju krožno povežemo še stolpce, nastane **torus**. Če je $p = q = N$, govorimo o N-torusu. Dodatne povezave (krožne vrstične in stolpične povezave) včasih speljemo preko več vrstic ali stolpcev (zavito krožno povezovanje). Če na primer v ravninski mreži zadnje vozlišče prve vrstice povežemo s prvim vozliščem druge vrstice, zadnje vozlišče druge vrstice s prvim v tretji, ..., zadnje vozlišče v zadnji vrstici s prvim v prvi vrstici, dobimo **zaviti valj**. S stolpičnim zavitim krožnim povezovanjem valja dobimo **zaviti torus**, zavitega valja pa **dvakrat zaviti torus**. Mreža je slabo razširljiva in dobro odporna na napake. Premer mreže je $p + q - 2$, N-torusa pa $2 * \lfloor N/2 \rfloor = \mathcal{O}(\sqrt{N})$.



Pomemben razred omrežij so k**-krožni grafi**, ki jih označimo z $G(n;\ \pm h_1, \ldots, \pm h_k)$ pri $0 < h_1 < h_2 < \cdots < h_k < \lfloor n/2 \rfloor$ in $h_j \in \mathbb{N}, j = 1, 2, \ldots, k$. Vozlišče i je povezano z vozlišči $i \pm h_1 (\text{mod}\, n)$, $i \pm h_2 (\text{mod}\, n)$, ..., $i \pm h_k (\text{mod}\, n)$. Omrežje je vozliščno simetrično, dobro odporno na napake, slabo razširljivo, premer pa je odvisen od števil n, h_1, $\cdots$, h_k. k**-krožni graf** je povezan natanko tedaj, ko $\gcd(n, h_1, h_2, \cdots, h_k) = 1$.



2.2 Usmerjevalni modeli

Komunikacija med delovnimi enotami v večprocesorskem sistemu ponavadi poteka z izmenjavo podatkov – sporočil. Sosednji vozlišči si lahko posredujeta sporočilo neposredno po skupni povezavi, zahtevnejša pa je izmenjava podatkov med vozlišči, ki med seboj niso neposredno povezana. Ta morajo, kadar želijo drug drugemu prenesti sporočilo, uporabiti tudi druga vozlišča omrežja. Pravimo, da sporočilo **potuje** po omrežju od izvora k cilju. **Usmerjevalni model** omrežja določa način izmenjave sporočil. V nadaljevanju bomo opisali tri modele: preklapljanje povezav, usmerjanje v omrežju z vodili in paketno usmerjanje ([2, 5, 34]).

Model **preklapljanja povezav** (*angl. circuit switching model*) je sestavljen iz treh faz: vzpostavitev zveze, prenos podatkov in rušenje zveze. V prvi fazi se vzpostavi celotna komunikacijska pot med izvorom in ciljem, povezave na tej poti se tako rezervirajo in jih do konca tretje faze (rušenje zveze) ne more uporabljati nihče drug. V drugi fazi potuje

sporočilo po vzpostavljeni poti, v zadnji fazi pa se pot ponovno razpusti in povezave so spet na voljo tudi drugim enotam.

Model **usmerjanja v omrežju z vodili** (*angl. routing along a network of buses*) se od preklapljanja povezav razlikuje po tem, da poti ne rezervira vnaprej, le algoritem, ki usmerja sporočilo, zagotavlja, da dve sporočili ne uporabita istega vodila hkrati.

2.2.1 Paketno usmerjanje

Pogosteje kot prejšnja dva modela pa se uporablja t.i. **paketno usmerjanje** (*angl. packet routing*) ([5, 34]). Sporočilo se preoblikuje v enega ali več paketov, vsak paket pa (včasih) v več manjših delov (*angl. flits*). Paket pozna svoj cilj, ki je enak cilju prvotnega sporočila. Usmerjevalni algoritem poskrbi, da vsi paketi pridejo do cilja, kjer se ponovno združijo v prvotno sporočilo. Usmerjanje poteka v več zaporednih *usmerjevalnih korakih*. V vsakem koraku lahko paket naredi en skok (*angl. hop*) , to pomeni, da se lahko premakne iz vozlišča, kjer se trenutno nahaja, v eno svojih sosed. Paketi se po omrežju premikajo med seboj povsem neodvisno, deli paketa pa nikoli ne potujejo ločeno, saj le prvi del (glava) ve za cilj, drugi deli mu preprosto sledijo.

Shrani-in-pošlji usmerjanje
Osnovni in tudi najpogostejši model paketnega usmerjenja, pri katerem ima vsako vozlišče možnost shranjevanja določenega števila paketov in pošiljanja teh svojim sosedam, se imenuje **shrani-in-pošlji** (*angl. store-and-forward*) [2, 14, 34], včasih tudi model preklapljanja paketov (*angl. packet switching*). Vozlišče, ki sprejema paket, mora počakati, da se ta ves zbere v njem, šele nato ga lahko pošlje naprej. Model je učinkovit, ko je trajanje prenosa paketa med sosednjima vozliščema omrežja neodvisno od (fizične) razdalje med njima. V tem primeru pravimo, da uporabljamo t.i. **konstantni** model. Ta se razlikuje od **linearnega** modela, pri katerem je trajanje t prenosa paketa linearna funkcija $t = \alpha + b\beta$, kjer je α začetni čas, b velikost paketa in β hitrost prenosa [34].

Verižno usmerjanje
Pri modelu shrani-in-pošlji potuje paket od začetnega vozlišča proti končnemu skozi več vmesnih vozlišč. V vsakem od njih počaka, da vanj pride v celoti in šele nato nadaljuje svojo pot. V linearnem modelu je ta način prenosa zamuden, saj bi lahko tisti deli paketa, ki so že prišli v vmesno vozlišče, nadaljevali pot brez čakanja na druge dele. Model, kjer "glava" paketa nadaljuje svojo pot, čeprav "rep" paketa še ni prispel v vozlišče, imenujemo **verižni model usmerjanja** (*angl. wormhole routing*) [2, 14, 34, 44]. Celoten paket se na začetku razdeli v manjše dele (*angl. flits*), ki samostojno, toda v koloni (drug za drugim) prodirajo proti cilju. Paket je lahko v nekem hipu (po delih) v več vozliščih omrežja, toda sosednji deli paketa so med prenosom vedno bodisi v istem bodisi v sosednjih vozliščih. Največja težava omenjene metode je nevarnost t.i. **mrtvih zapor** (*angl. deadlock*), kjer deli paketov ciklično ovirajo drug drugega pri napredovanju. Mrtve zapore lahko preprečimo z dovolj dolgimi čakalnimi vrstami in pravilnimi postopki

usmerjanja. Poseben primer verižnega usmerjanja, pri katerem do mrtvih zapor ne more priti, se imenuje **prodorno** (*angl. cut-through*) usmerjanje.

2.2.2 Tip povezave

Pogosta predpostavka usmerjevalnih modelov je, da sme po povezavi hkrati potovati le en paket oziroma del paketa. Zaradi omejitev, ki jih narekuje topologija omrežja, se modeli razlikujejo v razlagi zgornje predpostavke. Nekateri modeli dovoljujejo hkraten prenos paketov v nasprotnih smereh (še vedno pa v vsaki smeri največ en paket). V tem primeru govorimo o **hkratnem dvosmernem** (*angl. full-duplex*) modelu usmerjanja (opomba: izraz full-duplex se v slovenščino včasih prevaja tudi kot polni dupleka, full-duplex model pa kot polno dvojno povezan model). **Neusmerjeni** model sicer dovoljuje dvosmerni prenos, vendar ne sočasen (t.j. povezava v nekem koraku prepušča le v eno smer), **usmerjeni** model pa predvideva, da so povezave enosmerne in vedno omogočajo prenos v isti smeri.

2.2.3 Stopnja vzporednosti

Glede na stopnjo vzporednosti ločimo dva tipa usmerjevalnih modelov [34]. Pri modelu **SIMD** je en usmerjevalni korak omejen na eno samo smer, kar pomeni, da lahko vozlišče v enem koraku pošlje samo en paket po eni od povezav. Pri modelu **MIMD** pa lahko vozlišče pošlje pakete več svojim sosedam, seveda ob upoštevanju omenjenih omejitev in morebitne zasedenosti povezav.

2.2.4 Reševanje trkov

Če sporočila potujejo proti cilju brez vnaprejšnje rezervacije poti, se lahko zgodi, da v nekem koraku dve sporočili zahtevata isti vir – želita na primer potovati po isti povezavi. Pravimo, da je prišlo do **trka**. Sporočili se morata nekako dogovoriti, katero od njiju bo v tej bitki zmagovalec, katero bo torej dobilo vir, za katerega se oba potegujeta. Zmagovalec uporabi željeni vir, vprašanje pa je, kaj se zgodi s poraženčem. Ločimo štiri načine reševanja omenjenega problema: skladiščenje, blokiranje, odmetavanje in preusmerjanje [2]. **Skladiščenje** (*angl. buffering*) je pogosto uporabljan način: poražena sporočila se shranijo v čakalno vrsto, kjer čakajo, da se zasedeni vir sprosti. Težava, ki jo pomeni skladiščenje, je v velikih spominskih zmogljivostih, potrebnih za realizacijo čakalnih vrst. Skladiščenje se uporablja pri modelu shrani-in-pošlji in pri prodornem usmerjanju. **Blokiranje** (*angl. blocking*) je način, ko poraženo sporočilo preprosto obstane na mestu trka in počaka na prosti vir, pri tem pa ne sprosti virov, ki jih trenutno zaseda. V omrežjih z gostim prometom nesproščeni viri povzročijo dodatne težave, zato lahko pride do mrtvih zapor. Blokiranje se uporablja pri verižnem usmerjanju. Najslabši način, ki se v praksi uporablja zgolj kot izhod v sili, je **odmetavanje** (*angl. dropping*); poraženo sporočilo je preprosto zavrženo in nikoli ne pride na cilj. Odmetavanje se še posebej slabo obnese v omrežjih z gostim prometom, kjer raje uporabimo **preusmerjanje** (*angl. misrouting*).

Poraženo sporočilo pošljemo po drugi, morda napačni poti. Če ne bo mrtvih zapor, bo tudi to prišlo na cilj.

2.3 Usmerjevalni problemi

Problem paketne izmenjave podatkov lahko na splošno opišemo takole: naj bo dano omrežje z n vozlišči, vsako naj vsebuje nič ali več paketov in naj bo za vsak paket znano njegovo ciljno vozlišče. Poiskati želimo usmerjevalni algoritem, to je postopek, ki bo poskrbel, da bo vsak paket prispel na cilj, skladno z omejitvami, ki jih nalagata topologija omrežja in izbrani usmerjevalni model [2, 34, 44, 53].
Najpreprostejši je primer, ko ima vsak paket natanko določen (en!) cilj, in pri katerem je v vsako vozlišče namenjen kvečjemu en paket [34]. Omenjen problem imenujemo tudi problem **eden-enemu** (*angl. one-to-one*). Usmerjevalni problem **več-enemu** (*angl. many-to-one*) je problem, pri katerem ima lahko več paketov isti cilj, **eden-večim** (*angl. one-to-many*) pa problem, pri katerem ima en paket lahko več ciljev, oziroma pri katerem moramo več kopij istega paketa poslati različnim ciljnim vozliščem.
V najbolj splošnem primeru lahko med postopkom usmerjanja nastajajo tudi novi paketi. V tem primeru pravimo, da gre za problem **sprotnega** (*angl. on-line*) usmerjanja. Primer, ko paketi nimajo vnaprej določenega cilja (ta se izračuna sproti med izvajanjem usmerjanja) pa je pravzaprav poseben primer pravkar omenjenega sprotnega usmerjanja, saj pakete brez cilja lahko razumemo kot dodatne parametre, na podlagi katerih nastanejo novi paketi z natančno določenim ciljem.

2.3.1 Splošni usmerjevalni problem

Usmerjevalne probleme, kjer je število paketov natančno določeno že pred začetkom usmerjanja (torej paketi med samim postopkom ne morejo nastajati) in kjer imajo paketi že v začetku natančno določene cilje, imenujemo **splošni usmerjevalni problem**. Včasih ga imenujemo tudi (N, p, k_1, k_2)-usmerjevalni problem, kjer N pomeni število paketov, ki so na začetku v p vozliščih; na začetku v nobenem izmed vozlišč ni več kot k_1 paketov, na koncu pa v nobenem od ciljnih vozlišč ni več kot k_2 paketov. Splošni usmerjevalni problem zajema veliko problemov, ki jih delimo v več razredov.

Več vozlišč – isto sporočilo
V prvem razredu so usmerjevalni problemi, pri katerih več vozlišč oddaja ali sprejema isto sporočilo, torej gre za usmerjevalne probleme eden-večim in več-enemu. Pri **enovozliščnem oddajanju** (*angl. single node broadcast*) je eno vozlišče oddajnik, ki vsem drugim vozliščem pošlje isto sporočilo. Pri **enovozliščnem sporočanju** (*angl. single node scatter*) eno vozlišče odda vsem ostalim po eno sporočilo, vendar so zdaj sporočila med seboj različna. Pri **večvozliščnem oddajanju** (*angl. multi-node broadcast*) vsako od n vozlišč odda po eno sporočilo vsem drugim vozliščem; gre torej za sočasno n-kratno enovozliščno oddajanje, v omrežju je n različnih sporočil. **Večvozliščno sporočanje** oziroma

čvekanje (*angl. total exchange, gossiping*) je sočasno n-kratno enovozliščno sporočanje, v omrežju je n^2 različnih sporočil. **Enovozliščno zbiranje** (*angl. single node accumulate, gather*) je nasprotna operacija enovozliščnemu sporočanju, **večvozliščno zbiranje** (*angl. multi-node accumulate*) pa nasprotna operacija večvozliščnemu oddajanju.

Permutacijski usmerjevalni problemi
Razred problemov, pri katerih je na začetku v vsakem vozlišču x natanko en paket, ki je namenjen v vozlišče $\pi(x)$, pri čemer je π permutacija množice vozlišč danega omrežja, imenujemo razred **permutacijskih usmerjevalnih problemov**. Gre torej za $(n, n, 1, 1)$-usmerjevalni problem na omrežju z n vozlišči. Problemi tega razreda se dodatno delijo glede na vrsto permutacije, med katerimi so najbolj znane BPC (*angl. bit permute and complement*), Omega in Inverzna Omega. **Delni permutacijski** usmerjevalni problem se od permutacijskega razlikuje po tem, da je v omrežju manj kot n paketov, preslikava π, ki določa cilj paketov, pa je injektivna preslikava. V skrajnem primeru, ko je v omrežju le en paket, govorimo o **osnovnem** (*angl. two terminal*) usmerjevalnem problemu. Preslikava π je v tem primeru transpozicija.

Izotonični problemi
V tretji razred – razred **izotoničnih problemov** – sodijo usmerjevalni problemi, pri katerih se razmerja med celoštevilskimi reprezentacijami izvorov in ciljev paketov ohranjajo ali pa spreminjajo z natančno določenimi (matematičnimi) pravili. **Delna skrčitev** (*angl. semicontraction*) je problem, pri katerem se razdalja med izvoroma in ciljema poljubnih dveh paketov zmanjša. Z drugimi besedami, razlika celoštevilskih reprezentacij ciljnih točk paketov iz točk x in y je manjša od razlike celoštevilskih reprezentacij omenjenih točk, oziroma $|\pi(x) - \pi(y)| \leq |x - y|$. Dualni problem k delni skrčitvi se imenuje **delna razširitev** (*angl. semiexpansion*). **Monotono** usmerjanje vključuje vse probleme, pri katerih se ohranja relativni vrstni red paketov. Posebna primera monotonega usmerjanja sta zbiranje in razprševanje. **Zbiranje** (*angl. packing*) vključuje usmerjevalne probleme, pri katerih se vsi paketi na koncu zberejo v vozliščih, katerih celoštevilske reprezentacije pripadajo nekemu intervalu (na primer prvih M vozlišč omrežja), poleg tega pa se ohrani njihov relativni vrstni red: za poljubna paketa, ki sta na začetku v vozliščih x in y, za kateri velja $x < y$, velja $\pi(x) < \pi(y)$. Dualni problem zbiranja se imenuje **razprševanje** (*angl. unpacking, spreading*). Vsak problem monotonega usmerjanja lahko rešimo z zbiranjem in razprševanjem: najprej pakete zberemo v nekem intervalu, nato jih razpršimo na ciljno mesto. Zbiranje in delna skrčitev ter njuna dualna problema so posebni primeri delnega permutacijskega usmerjanja.

Z uporabo razprševanja lahko vsak problem eden-večim prevedemo v problem eden-enemu. Pri reševanju usmerjevalnih problemov se torej lahko osredotočimo na probleme, pri katerih ima vsak paket natančno določen (en) cilj, saj lahko druge probleme rešimo v dveh korakih, najprej z razprševanjem, nato z reševanjem problema eden-enemu.

2.4 Usmerjevalni algoritmi

Za reševanje konkretnega usmerjevalnega problema je običajno na voljo več pristopov. Katerega od njih bomo izbrali, je odvisno predvsem od usmerjevalnega modela, od topologije omrežja in od narave problema. Od izbranega pristopa oziroma od njegove implementacije v obliki algoritma ponavadi pričakujemo, da bo problem rešil v čim krajšem času, to je, da bo vse pakete pripeljal do cilja v čim manj korakih [5, 14, 34, 44].

Algoritme za reševanje usmerjevalnih problemov delimo v dve skupini: **dinamični** (*angl. on-line*) in **statični** (*angl. off-line*) algoritmi. Statični algoritmi pred začetkom izvajanja za vsak paket izračunajo pot, po kateri bo potoval, nato v zaporednih korakih pakete po izračunani poti privedejo na cilj. Pri računanju uporabljajo podatke o vseh paketih in o celotnem omrežju. Dinamični algoritmi, ki se dodatno delijo na pozabljive in prilagodljive, pakete usmerjajo sproti, torej na vsakem usmerjevalnem koraku za vsak paket izračunajo naslednji korak na njegovi poti. Pri računanju uporabljajo samo lokalne informacije o paketu, o vozlišču, v katerem je paket, in o sosednjih vozliščih. **Deterministično pozabljivi** (*angl. deterministic oblivious*) algoritmi usmerjajo paket le na podlagi podatkov o začetnem in končnem vozlišču, **naključno pozabljivi** (*angl. randomized oblivious*) pa pri usmerjanju dodatno uporabljajo neko naključno zaporedje bitov. **Prilagodljivi** (*angl. adaptive*) algoritmi lahko poleg podatkov o ciljnem vozlišču uporabljajo tudi podatke o trenutnem položaju paketa. Slabost pozabljivih algoritmov je predvsem, da zaradi nepoznavanja podatkov o trenutnem stanju (razporeditev paketov), lahko pride do **mrtvih zapor** (*angl. deadlock*), torej do cikličnih, med seboj neodvisnih zahtev za dane vire (povezave in čakalne vrste). Mrtve zapore lahko preprečimo z uporabo prilagodljivih algoritmov, ki se znajo sproti odločiti za nadaljevanje ali pa kroženje preprosto preprečijo z naključnim usmerjanjem paketov. Algoritem, ki uporablja omenjeno, popolnoma naključno strategijo, se imenuje **algoritem norega poštarja** (*angl. mad-postman*).

Algoritem vročega krompirja (*angl. hot-potato*) ne uporablja čakalnih vrst. Ko paket prispe v neko vmesno vozlišče, takoj nadaljuje pot; če je povezava, ki paket najhitreje vodi proti cilju, trenutno zasedena, bo paket nadaljeval pot v kaki drugi smeri. Čakalne vrste, ki vozlišču omogočijo, da hrani pakete toliko časa, da pridejo na vrsto za pametno nadaljevanje poti, so za uspešno usmerjanje zelo pomembne. Po drugi strani pa dolge čakalne vrste pomenijo manjšo zanesljivost sistema in zahtevnejšo realizacijo (težave s pomnilnikom), zato algoritme ponavadi načrtujejo tako, da ti uporabljajo najkrajše možne čakalne vrste, ki še zagotavljajo njihovo učinkovitost.

Izvajanje nekaterih algoritmov privede pakete v t.i. **žive zapore** (*angl. livelock*): paketi sicer potujejo, toda ciklično se vrtijo v zankah in ne napredujejo proti cilju.

Verjetnostni usmerjevalni algoritmi (*angl. probabilistic routing*) uporabljajo zaporedje naključnih bitov za pretvarjanje tistih porazdelitev vhodnih podatkov, pri katerih nastopajo primeri z najslabšo časovno zahtevnostjo, v obvladljivo in predvidljivo porazdelitev. Omenjeno tehniko uporabljamo v primerih, ko je najslabši primer vhodnih podatkov za deterministični algoritem bistveno slabši od povprečnega. Kljub obetavnim možnostim

pa obstaja domneva, da ima verjetnostni usmerjevalni algoritem isto časovno zahtevnost kot primerno izdelan dinamično prilagodljiv algoritem.
Poleg omenjenih pristopov lahko problem usmerjanja rešimo tudi s prevedbo (*angl. reduction*) v drug problem, katerega rešitev poznamo. Tako lahko na primer problem permutacijskega usmerjanja prevedemo v problem urejanja, problem delnega permutacijskega usmerjanja pa v problem urejanja in razprševanja.

2.5 Ocenjevanje kakovosti algoritmov

Kakovost usmerjevalnih algoritmov lahko ocenjujemo na več različnih načinov, odvisno od namena in načina uporabe usmerjevalnega algoritma ter od razpoložljivih virov. Tako lahko kakovost algoritma ocenjujemo glede na prostorsko zahtevnost (koliko dodatnega pomnilnika potrebuje pri reševanju usmerjevalnih problemov) ali pa glede na časovno zahtevnost (koliko usmerjevalnih korakov naredi pri reševanju danega problema; koliko korakov je nepotrebnih; koliko časa v povprečju čaka posamezen paket in podobno). V določenih primerih nas zanima le, ali je algoritem pakete vodil po najkrajših poteh in ali je vse pakete pripeljal na pravi cilj. Običajno največ pozornosti posvetimo tistemu kriteriju, ki se nanaša na najpomembnejši (najdragocenejši) vir. V našem primeru je to čas, ki ga algoritem potrebuje, da reši dani usmerjevalni problem, ta pa je neposredno odvisen od števila usmerjevalnih korakov, ki jih pri tem opravi. V tem delu smo zato kakovost algoritma $\mathcal{A}$ ocenjevali s številom $K(\mathcal{A})$, ki je povprečni kvocient med številom opravljenih korakov in dolžino najdaljše poti konkretnega usmerjanja. Natančneje je število $K(\mathcal{A})$ definirano, kot sledi.
Naj bo $\mathcal{G}$ množica vseh krožnih grafov. Za $G \in \mathcal{G}$ bomo s *splošni usmerjevalni problem za graf* G imenovali vsako začetno razporeditev paketov z natančno določenenimi cilji. Množico vseh splošnih usmerjevalnih problemov za graf G bomo označevali s $\mathcal{P}_G$.
Naj bo $G \in \mathcal{G}$ in $p \in \mathcal{P}_G$. S koraki$(\mathcal{A}, G, p)$ označimo število usmerjevalnih korakov, ki jih potrebuje algoritem $\mathcal{A}$ pri reševanju problema p na grafu G, z $\ell(G, p)$ pa največjo razdaljo med začetnim in končnim vozliščem paketov problema p. Jasno je, da koraki$(\mathcal{A}, G, p) \geq \ell(G, p)$, saj najbolj oddaljen paket potrebuje najmanj $\ell(G, p)$ korakov, da pride na cilj. Za ocenjevanje kakovosti reševanja konkretnega usmerjevalnega problema p z algoritmom $\mathcal{A}$ definirajmo število $K(\mathcal{A}, G, p)$ s

$$K(\mathcal{A}, G, p) = \frac{\text{koraki}(\mathcal{A}, G, p)}{\ell(G, p)}.$$

Število $K(\mathcal{A}, G, p)$ je vedno večje ali (v optimalnem primeru) kvečjemu enako 1. Bolj ko se v konkretnem primeru približamo vrednosti 1, bolj uspešno je usmerjanje.

Kakovost usmerjevalnega algoritma $\mathcal{A}$ bi sedaj lahko definirali kot limito povprečnih vrednosti števil $K(\mathcal{A}, G, p)$, ko G preteče vso množico $\mathcal{G}$ in, pri posameznem G, p vso množico $\mathcal{P}_G$. Pri tem bi bilo treba natančneje opisati, kako naj omenjeno limito izračunamo, saj je od tega morda odvisna tudi njena končna vrednost.

Pri našem delu smo namesto natančne vrednosti povprečja števil $K(\mathcal{A}, G, p)$ uporabljali njegov približek. Ta je označen s $K_{(\mathcal{G}',\mathcal{P})}(\mathcal{A})$ in definiran kot

$$K_{(\mathcal{G}',\mathcal{P})}(\mathcal{A}) = AVG_{G\in\mathcal{G}',p\in\mathcal{P}(G)}K(\mathcal{A}, G, p),$$

pri čemer je $\mathcal{G}' \subset \mathcal{G}$ končna množica, $\mathcal{P}$ pravilo, ki za posamezen graf $G \in \mathcal{G}'$ podaja končno množico usmerjevalnih problemov $\mathcal{P}(G) \subset \mathcal{P}_G$, AVG pa funkcija, ki sešteje vse vrednosti $K(\mathcal{A}, G, p)$ ter seštevek deli z njihovim številom. Kadar je iz konteksta razvidno, katero podmnožico $\mathcal{G}'$ in kakšno pravilo $\mathcal{P}$ uporabljamo, število $K_{(\mathcal{G}',\mathcal{P})}(\mathcal{A})$ označimo preprosto s $K(\mathcal{A})$.
Ker je število $K_{(\mathcal{G}',\mathcal{P})}(\mathcal{A})$ odvisno od izbire podmnožice $\mathcal{G}'$ ter pravila $\mathcal{P}$ in zato ni primerno za ocenjevanje absolutne kakovosti posameznega algoritma, smo ga v nadaljevanju tega dela uporabljali zgolj kot merilo za primerjavo kakovosti dveh usmerjevalnih algoritmov pri istem $\mathcal{G}'$ in $\mathcal{P}$.

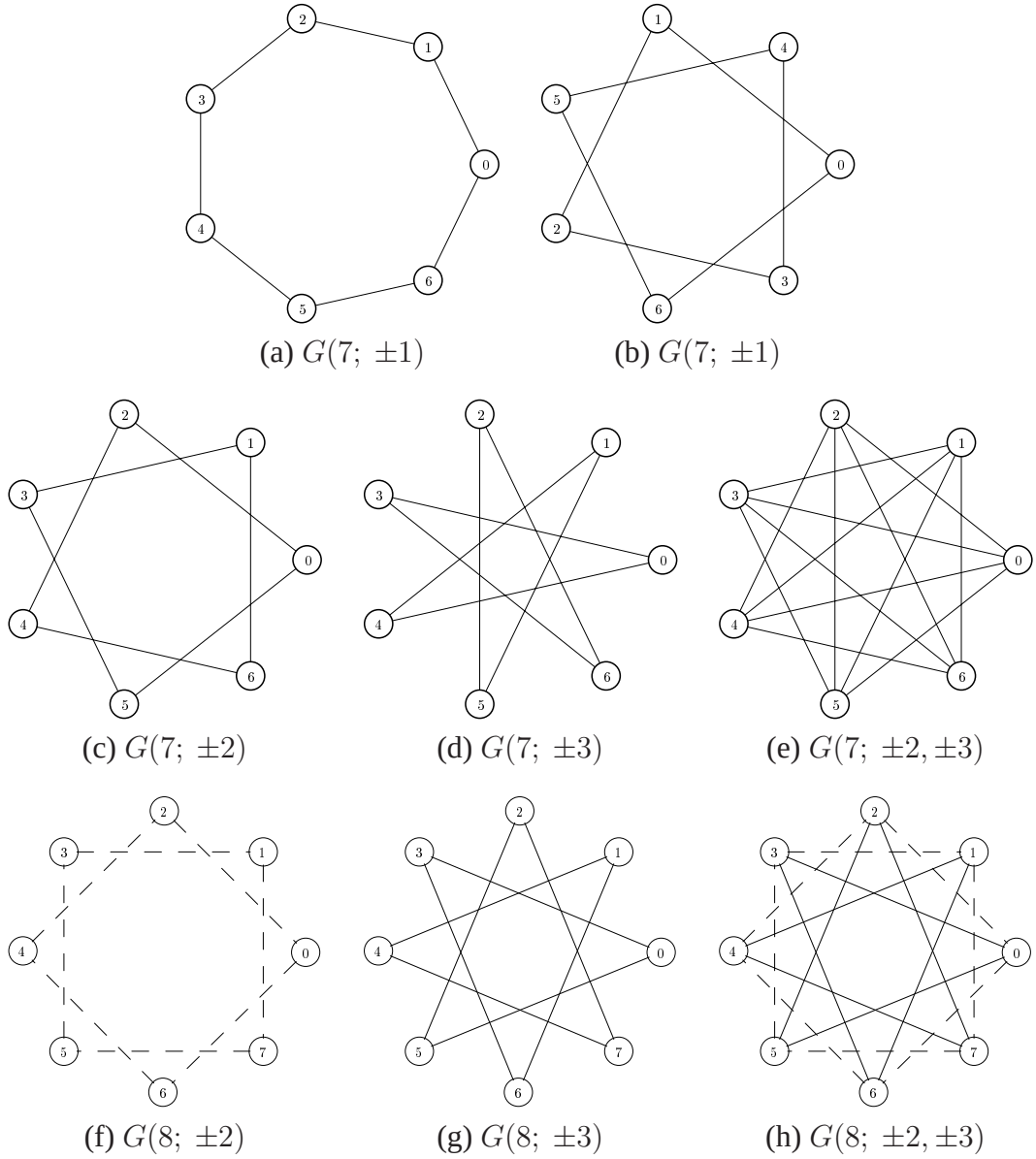
Poglavje 3

Krožni grafi

Pričujoče poglavje je namenjeno natančni predstavitvi krožnih grafov, ki predstavljajo veliko družino simetričnih grafov, v kateri so poleg drugih tudi cikli, polni grafi in nekateri torusi. Na začetku poglavja bomo najprej pojem krožnega grafa uvedli na neformalen način, nato bomo podali štiri različne definicije in pokazali njihovo ekvivalentnost. Predstavili bomo najpomembnejše topološke lastnosti krožnih grafov ter način, kako z uporabo celoštevilske mreže rešujemo nekatere probleme, kot sta, na primer, problem računanja premera in problem iskanja najkrajše poti med danim parom točk. Pri tem bomo veliko pozornosti posvetili pojmom, kot so celoštevilska mreža, modul, pakirana baza, množica X_w ter njeni predstavniki in podobno. Uvedli bomo pojem delno usmerjenih krožnih grafov ter v zadnjem razdeleku poglavja pokazali, kako z uporabo minimalne projekcije izračunamo najkrajše poti v teh grafih. Pri dokazovanju pravilnosti omenjenega algoritma bomo uporabili metode, leme in trditve, ki so predstavljene v prvih razdelkih tega poglavja.
Pri svojem delu smo se bolj kot s splošnimi k-krožnimi grafi ukvarjali z družino dvokrožnih grafov, zato je njim namenjeno nekoliko več pozornosti tudi v tem poglavju. Ker nekaterih pojmov, povezanih z reševanjem problemov v dvokrožnih grafih, ne znamo posplošiti, se bodo nekateri rezultati v nadaljevanju nanašali zgolj na dvokrožne grafe; kjer je posplošitev znana, jo bomo omenili.

3.1 Kaj je krožen graf?

Preden se lotimo natančne predstavitve krožnih grafov s formalnimi definicijami, poglejmo, kako bi jih opisali na neformalen način. Že samo ime da slutiti, da k-krožne grafe sestavlja k ponovitev neke krožne (ciklične) strukture. Izkaže se, da je pod določenimi pogoji (o teh pogojih malo kasneje) ta predstava povsem na mestu, saj je 1-krožen graf pravzaprav cikel, dvokrožen graf preplet dveh ciklov in k-krožen graf preplet k ciklov. Na slikah 3.1-I(a) in 3.1-I(b) je na dva različna načina prikazan najpreprostejši primer krožnega grafa, 1-krožen graf $G(7;\ \pm 1)$. Razlika med omenjenima grafoma je le v razporeditvi točk v ravnini; grafa 3.1-I(a) in 3.1-I(b) sta izomorfna, saj oba predstavljata cikel dolžine 7. Čeprav gre pri omenjenih grafih za dva različna grafična prikaza istega



(a) $G(7;\ \pm 1)$ (b) $G(7;\ \pm 1)$

(c) $G(7;\ \pm 2)$ (d) $G(7;\ \pm 3)$ (e) $G(7;\ \pm 2, \pm 3)$

(f) $G(8;\ \pm 2)$ (g) $G(8;\ \pm 3)$ (h) $G(8;\ \pm 2, \pm 3)$

Slika 3.1-I: Primer preprostih krožnih grafov $G(7;\ \pm 1)$ $G(7;\ \pm 2)$ $G(7;\ \pm 3)$ $G(7;\ \pm 2, \pm 3)$ $G(8;\ \pm 2)$ $G(8;\ \pm 3)$ $G(8;\ \pm 2, \pm 3)$

grafa, se na prvi pogled zdi, da gre za dva različna grafa. Točke grafa na sliki 3.1-I(b) niso povezane z direktnimi sosedami (glede na razporeditev točk v ravnini) ampak s sosedami sosed. S preimenovanjem točk grafa na sliki 3.1-I(b) (točke označimo od 0 do 6 po vrsti, kot si sledijo v obratni smeri urinega kazalca) dobimo grafu $G(7;\ \pm 1)$ izomorfen 1-krožni graf $G(7;\ \pm 2)$ (glej sliko 3.1-I(c)). V tem grafu je točka 0 povezana s točko 2, točka 1 s točko 3, točka 2 s točko 4 in tako naprej. Podobno graf na sedmih točkah, v katerem je vsaka točka povezana s sosedo sosedove sosede (glede na razporeditev točk v ravnini) označimo z $G(7;\ \pm 3)$ (slika 3.1-I(d)). Grafi na slikah 3.1-I(a), 3.1-I(c) in 3.1-I(d) predstavljajo tri različne 1-krožne grafe na sedmih točkah. Dejstva, da sta $G(7;\ \pm 2)$ in $G(7;\ \pm 3)$ izomorfna grafu $G(7;\ \pm 1)$, ne moremo posplošiti na poljubne 1-krožne grafe. Že pri $G(8;\ \pm 2)$ namreč vidimo, da ta razpade na dva cikla dolžine 4 (slika 3.1-I(f)) in torej ni izomorfen ciklu $G(8;\ \pm 1)$. Ni se težko prepričati, da je graf $G(n;\ \pm h)$ izomorfen grafu $G(n;\ \pm 1)$ tedaj in le tedaj, ko velja $\gcd(n, h) = 1$.

Podobno kot lahko (malo površno) za 1-krožne grafe rečemo, da so to cikli dolžine n, lahko za dvokrožne grafe, z isto mero nenatančnosti, rečemo, da so to grafi, ki jih sestavljata dva prepletena cikla. Primer dveh prepletenih ciklov vidimo na sliki 3.1-I(e), ki prikazuje graf $G(7;\ \pm 2, \pm 3)$. Ta graf dobimo tako, da točke 0, 1, 2, ..., 6 povežemo s povezavami grafov $G(7;\ \pm 2)$ in $G(7;\ \pm 3)$ (graf $G(7;\ \pm 2, \pm 3)$ grafično dobimo tako, da grafa $G(7;\ \pm 2)$ in $G(7;\ \pm 3)$ narišemo enega preko drugega). Nenatančnost zgornje definicije dvokrožnih grafov se odraža že v naslednjem primeru. Graf $G(8;\ \pm 2, \pm 3)$ (slika 3.1-I(h)) je namesto iz dveh sestavljen iz treh ciklov – dva cikla dolžine 4 "prispeva" graf $G(8;\ \pm 2)$, en cikel dolžine 8 pa graf $G(8;\ \pm 3)$. Do "drobitve" ciklov je prišlo, ker števili 8 in 2 nista tuji. Ob upoštevanju te ugotovitve lahko površen zgornji opis delno popravimo in zapišemo, da je dvokrožni graf $G(n;\ \pm h_1, \pm h_2)$ sestavljen iz $\gcd(n, h_1)$ ciklov dolžine $n/\gcd(n, h_1)$ in $\gcd(n, h_2)$ ciklov dolžine $n/\gcd(n, h_2)$. Podobno za splošen k-krožni graf $G(n;\ \pm h_1, \ldots, \pm h_k)$ ne moremo reči, da je sestavljen iz k ciklov dolžine n, pač pa lahko rečemo, da vsak $i \in \{1, 2, \ldots, k\}$ v njem prispeva $\gcd(n, h_i)$ ciklov dolžine $n/\gcd(n, h_i)$.

Poleg števila točk n in dolžine povezav h_i za $i \in \{1, 2, \ldots, k\}$ je pomembna lastnost krožnih grafov tudi tip povezav. Ločimo med usmerjenimi in neusmerjenimi povezavami ter, v odvisnosti od uporabljenih povezav, med usmerjenimi, neusmerjenimi in delno usmerjenimi krožnimi grafi. Definicija prvih dveh je razvidna že iz imena (prvi ima vse povezave usmerjene, drugi pa neusmerjene), tretjega, delno usmerjenega, pa opišemo kot graf, v katerem so nekatere povezave usmerjene, druge pa neusmerjene (za podrobnosti glej razdelek 3.12). Morda ni odveč, če poudarimo, da je stopnja posamezne točke in skupno število povezav v vseh treh primerih enako, razlika je v funkciji, ki jo posamezna povezava v grafu opravlja. V usmerjenem k-krožnem grafu ima tako vsaka točka k vhodnih in k izhodnih povezav, medtem ko ima v neusmerjenem primeru $2k$ vhodnih in $2k$ izhodnih. Usmerjen k-krožen graf (*angl. k-loop graph*) označimo z $G(n;\ h_1, \ldots, h_k)$, neusmerjen k-krožen graf (*angl. k-circulant graph*) pa z $G(n;\ \pm h_1, \ldots, \pm h_k)$.

3.2 Definicije in dokaz njihove ekvivalentnosti

V literaturi so krožni grafi definirani na več načinov, ki so, če so le dovolj natančni, med seboj ekvivalentni (to je, vsi opisujejo isto družino grafov). Zaradi te "neenotnosti" skoraj ne najdemo dveh različnih del, v katerih bi bili krožni grafi podani na povsem enak način. Da bi bodočim bralcem olajšali branje literature o krožnih grafih, smo na tem mestu zbrali in poenotili najbolj pogosto uporabljene definicije ter dokazali njihovo ekvivalentnost.
Uvedimo najprej nekatere skupne oznake. Naj bodo n, k, $h_1, h_2, \ldots, h_k$ naravna števila. Nekateri avtorji v definiciji krožnih grafov predpostavljajo, da so števila h_i vsa manjša od n, mi pa smo postavili na videz precej strožji pogoj, in sicer

$$0 < h_1 < h_2 < \ldots < h_k < \left\lfloor \frac{n}{2} \right\rfloor .$$

Naše definicije bodo tako zajele le krožne grafe, katerih točke imajo stopnjo $2k$. V primerih namreč, ko je eno od števil h_i enako polovici vrednosti števila n, se povezavi $(i, i + n/2)$ ter $(i + n/2, i)$ združita v eno in stopnja točk postane liha $(2k - 1)$. Ker je teh posebnih primerov malo in ker onemogočajo enotno obravnavo krožnih grafov, smo se, tako kot večina avtorjev, odločili, da se jim že v definiciji ognemo.
Označimo z V_n množico nenegativnih celih števil, manjših od n,

$$V_n = \{0, 1, \ldots, n - 1\}.$$

Množica V_n, katere moč je n, bo v vseh primerih pomenila množico točk grafa. Z I^+ bomo označili množico vseh pozitivnih števil od 1 do k, z I^- množico vseh negativnih celih števil večjih ali enakih $-k$, z I pa unijo množic I^+ in I^-, torej

$$I^+ = \{1, 2, \ldots, k\}, \;\; I^- = \{-k, -k + 1, \ldots, -1\}, \;\; I = I^+ \cup I^-.$$

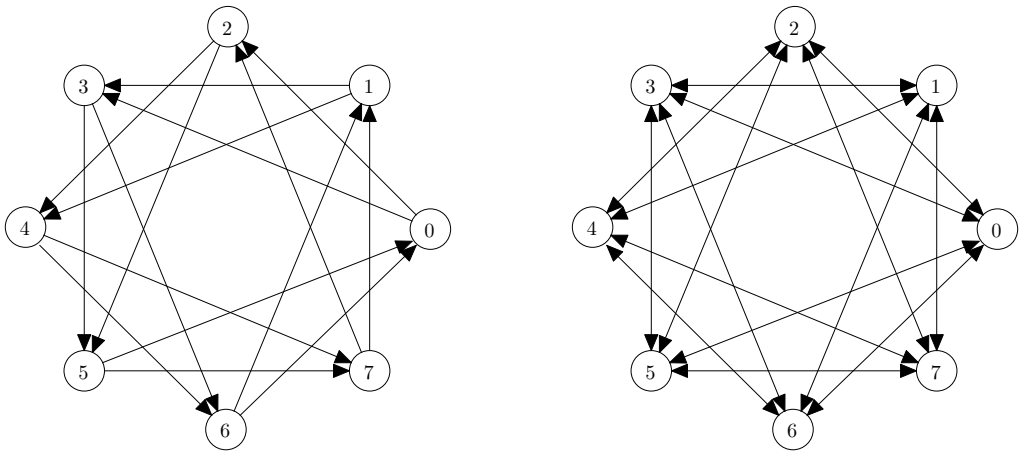
Krožni graf je natančno podan s števili n, k ter $h_1, \ldots, h_k$. Usmerjen k-krožni graf bomo označevali z $G(n;\ h_1, \ldots, h_k)$, njegovo neusmerjeno različico pa z $G(n;\ \pm h_1, \ldots, \pm h_k)$.

Prva definicija, ki smo jo zasledili v [6, 54], krožni graf poda s pravilom, ki določa sosednje pare. Relacijo sosednosti označuje s simbolom $\sim$, ki ima naslednji pomen: $u \sim v \iff$ iz točke u vodi povezava v točko v. Naj omenimo, da relacija sosednosti ni nujno simetrična – iz $u \sim v$ torej ne sledi nujno $v \sim u$ (glej primer 3.2-I). V definiciji bomo, za vsak $K \subset \mathbb{Z}$, graf nad množico točk V_n, v katerem povezave med točkami določa pravilo

$$\forall u, v \in V_n : u \sim v \iff \exists s \in K : v = u + s \pmod{n},$$

označili z $G(n, K)$.

Definicija KGA. *Naj bo* $S^+ = \{h_1, h_2, \ldots, h_k\}$ *in* $S^- = \{-h_1, -h_2, \ldots, -h_k\}$ *ter* $S = S^+ \cup S^-$. *Graf* $G(n;\ S^+)$ *imenujemo* usmerjen k-krožni graf *in označimo z* $G(n;\ h_1, \ldots, h_k)$, *graf* $G(n;\ S)$ *pa* neusmerjen k-krožni graf *in označimo z* $G(n;\ \pm h_1, \ldots, \pm h_k)$.



(a) $G(8;\ 2, 3)$ (b) $G(8;\ \pm 2, \pm 3)$

Slika 3.2-I: (a) Usmerjen in (b) neusmerjen krožni graf, $n = 8$, $k = 2$, $h_1 = 2$, $h_2 = 3$

Primer 3.2-I. Na sliki 3.2-I(a) je prikazan usmerjen graf $G(8;\ \{2, 3\})$. V njem je točka 4, na primer, povezana s točko 6, saj je $2 \in \{2, 3\}$ in velja $6 = 4 + 2 \pmod 8$, točka 6 pa ni povezana s točko 4, saj -2 ni element množice $\{2, 3\}$. V neusmerjeni različici tega grafa (slika 3.2-I(b)) pa za vsak par $u, v \in V_8$ velja: $u \sim v \iff v \sim u$, saj je za vsak s iz $\{-3, -2, 2, 3\}$ tudi $-s$ v tej množici.

Z naslednjo definicijo krožni graf podamo na bolj standarden način in sicer tako, da opišemo množico točk in množico povezav [1, 7, 8, 31, 41, 45, 48]. Razlika med usmerjeno in neusmerjeno različico grafa je v tem, da so pri usmerjenem povezave podane kot urejeni, pri neusmerjenem pa kot neurejeni pari točk.

Definicija KGB. *Naj bo množica urejenih parov E podana z*

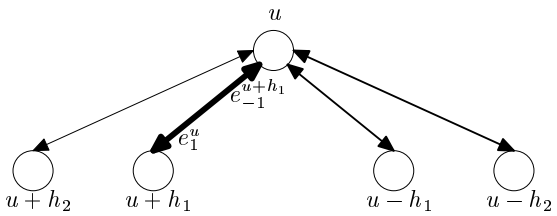
$$E = \{(u, u + h_i \ (\mathit{mod}\ n));\ u \in V_n;\ i \in \{1, 2, \ldots, k\}\}.$$

Graf z množico točk V_n in množico povezav E imenujemo usmerjen k-krožni graf. *Če v množici E urejene pare spremenimo v neurejene, se pripadajoči graf imenuje* neusmerjen k-krožni graf.

Iz te definicije neposredno sledi, da je v usmerjenem primeru vsaka točka u povezana s k sosedami in sicer s točkami $u + h_i \pmod n$, za vsak $i \in I^+$, manj pa je očitna naslednja lema.

Lema 3.2-I. *V neusmerjenem k-krožnem grafu $G(n;\ \pm h_1, \ldots, \pm h_k)$ je po definiciji KGB vsaka točka u povezana z $2k$ sosedami in sicer s točkami $u \pm h_i \ (\mathit{mod}\ n)$, za vsak $i \in I^+$.*

Dokaz. Vzemimo $u \in V_n$ in $i \in I^+$ in dokažimo, da je točka u povezana s točkama $u + h_i \pmod n$ in $u - h_i \pmod n$. Prvi del trditve drži, saj velja $(u, u+h_i \pmod n) \in E$, drugi del pa zato, ker za $v = u - h_i \pmod n$ velja $(v, v+h_i \pmod n) = (u - h_i \pmod n, u) \in E$. Ker je množica E množica neurejenih parov, rezultat sledi. □



Slika 3.2-II: Točka u neusmerjenega dvokrožnega grafa ter 4 sosednje točke

Povezave krožnega grafa lahko razdelimo v skupine: prvo skupino tvorijo tiste povezave, ki "pripadajo" številu h_1, drugo tiste, ki "pripadajo" številu h_2, in tako naprej. Skupna značilnost povezav ene skupine je fiksna razlika med pari točk, ki sestavljajo povezave. Tretja definicija krožnih grafov [4, 7, 36, 37, 39, 40] je zasnovana prav na tej lastnosti. Preden pa jo podamo, si oglejmo še, kako je formalno definiran posamezen tip povezav.

Definicija 3.2-II. *Za $i \in I$ elemente množice urejenih parov $\{(u, u + \textit{sign(i)} * h_{|i|} \ (\textit{mod}\ n);\ u \in V_n\}$ imenujemo* povezave tipa $\mathcal{E}_i$. *Povezavo tipa $\mathcal{E}_i$, ki se začne v točki $u \in V_n$, označimo z e_i^u.*

V usmerjenem krožnem grafu poznamo samo "pozitivne" povezave, to je povezave tipa $\mathcal{E}_i$, za $i \in I^+$, v neusmerjenem primeru pa so tem dodane še "negativne" povezave, torej povezave tipa $\mathcal{E}_i$, za $i \in I^-$.

Definicija KGC. *Za $i \in I$ označimo z E_i množico povezav tipa $\mathcal{E}_i$,*

$$E_i = \{(u, u + \textit{sign(i)} * h_{|i|} \ (\textit{mod}\ n));\ u \in V_n\}.$$

Graf z množico točk V_n in množico povezav $E = \bigcup_{i \in I^+} E_i$ imenujemo usmerjen k-krožni graf, *graf z množico točk V_n in množico povezav $E = \bigcup_{i \in I} E_i$ pa* neusmerjen k-krožni graf.

> Primer 3.2-II. V primeru neusmerjenih krožnih grafov povezava e_i^u hkrati nosi tudi oznako e_{-i}^v pri $v = u -$ sign$(i) * h_{|i|}$ (mod n) – ena in ista povezava namreč povezuje točki v dveh smereh. Na sliki 3.2-II je prikazana točka u dvokrožnega grafa ter 4 sosednje točke. Poudarjeno je prikazana povezava e_1^u, je hkrati tudi povezava $e_{-1}^{u+h_1}$.

Za predstavitev naslednje definicije krožnih grafov [6, 11], ki se precej razlikuje od prejšnjih treh, saj namesto z množico povezav E novo družino predstavi s pomočjo matrike sosednosti, potrebujemo pojem krožne matrike.

Definicija 3.2-III. *Matrika $A = (a_{i,j})_{i,j=0}^{n-1} \in \{0,1\}^{n*n}$ je* krožna matrika, *če je i-ta vrstica $(i \in \{1, \ldots n-1\})$ matrike A ciklični pomik ničte vrstice te matrike za i mest v desno.*

Primer 3.2-III. $A = \begin{Vmatrix} 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 \\ 1 & 0 & 0 & 0 & 1 \end{Vmatrix}$

Matrika A je krožna matrika. Naslednjo vrstico matrike dobimo s cikličnim premikom prejšnje vrstice v desno. Dve enici, ki sta v zgornji vrstici v skrajno levih stolpcih, se v vsaki vrstici pomakneta za eno mesto v desno.

Definicija KGD. *Naj bo $A = (a_{i,j})_{i,j=0}^{n-1} \in \{0,1\}^{n*n}$ krožna matrika z ničelno diagonalo. Če prva vrstica vsebuje natanko k enic, ki so v stolpcih z indeksi h_i ($a_{0,h_i} = 1$ za $i \in I^+$), se graf nad množico točk V_n, katerega matrika sosednosti je enaka matriki A, imenuje* usmerjen k-krožni graf $G(n;\ h_1, \ldots, h_k)$. *Če prva vrstica vsebuje natanko $2k$ enic in velja $a_{0,h_i} = 1$ za $i \in I^+$ ter je poleg tega simetrična ($a_{0,i} = a_{0,j}$ pri $j = -i \pmod n$), pripadajoči graf imenujemo* neusmerjen k-krožni graf $G(n;\ \pm h_1, \ldots, \pm h_k)$.

Opomba. Zaradi omejitev vrednosti števil h_i (vsa so manjša od polovice števila vozlišč) za prvo vrstico krožne matrike, ki pripada usmerjenemu krožnemu grafu, velja, da je vseh k enic zbranih v levi polovici te vrstice. Podobno je v prvi vrstici matrike, ki pripada neusmerjenemu krožnemu grafu, polovica enic v levi, polovica pa v desni strani vrstice.

Primer 3.2-IV. Krožnemu grafu $G(8;\ 2, 3)$ (slika 3.2-I(a)) pripada krožna matrika, katerega ničta vrstica je enaka $a_0 = (0, 0, 1, 1, 0, 0, 0, 0)$, grafu $G(8;\ \pm 2, \pm 3)$ (slika 3.2-I(b)) pa krožna matrika z $a_0 = (0, 0, 1, 1, 0, 1, 1, 0)$.

Čeprav se štiri definicije, ki smo jih predstavili, med seboj precej razlikujejo, vse opisujejo isto družino grafov. V to se prepričamo z naslednjo trditvijo.

Trditev 3.2-IV. *Definicije KGA, KGB, KGC in KGD opisujejo isto družino grafov.*

Dokaz te trditve je sestavljen iz dveh delov. V prvem bomo pokazali, da trditev drži za tisti del definicij, ki se nanašajo na usmerjene, v drugem pa se bomo osredotočili na neusmerjene grafe. Obakrat bomo ekvivalenco definicij dokazali s krožno implikacijo, in sicer v smeri KGA $\Rightarrow$ KGB $\Rightarrow$ KGC $\Rightarrow$ KGD $\Rightarrow$ KGA. Ker se množica točk grafa v vseh definicijah ujema, moramo pri dokazovanju posamezne implikacije preveriti le, ali so točke, ki so povezane po definiciji na levi strani implikacije, povezane tudi po tisti na desni strani.

Dokaz trditve 3.2-IV (za usmerjen primer).
KGA $\Rightarrow$ KGB. Naj v grafu $G(n;\ h_1, \ldots, h_k)$ iz točke u vodi povezava v točko v. Radi bi pokazali, da je potem par (u, v) element množice E (glej definicijo KGB). V to se zlahka prepričamo, saj iz $u \sim v$ po definiciji KGA sledi, da obstaja tak $s \in S^+$, da velja $v = u + s \pmod n$. Ker je množica S^+ sestavljena iz števil h_i, je s enak enemu izmed njih in zato velja: $v = u + h_i \pmod n$ za neki $i \in I^+$. Par (u, v) je torej enak paru $(u, u + h_i \pmod n)$, torej je element množice E. □

KGB $\Rightarrow$ KGC. Naj bo (u, v) par po definiciji KGB povezanih točk. Potem obstaja tak $i \in I^+$, da $(u, v) = (u, u + h_i \pmod n)$. Par (u, v) je torej element množice E_i in zato element unije $\cup_{i\in I^+} E_i$. □

KGC $\Rightarrow$ KGD. Vzemimo par po definiciji KGC povezanih točk (u, v). Ker je $(u, v) \in \cup_{i\in I^+} E_i$, obstaja tak $i \in I^+$, da je $(u, v) \in E_i$. Par (u, v), ki se zato lahko zapiše v obliki $(u, v) = (u, u + h_i \pmod n)$, je povezan tudi po definiciji KGD, saj velja $a_{0,h_i} = 1$ in, ker je u-ta vrstica u-kratni ciklični pomik ničte vrstice, $a_{u,u+h_i \pmod n} = 1$. □

KGD $\Rightarrow$ KGA. Če iz točke u po definiciji KGD vodi povezava v točko v, potem velja $a_{u,v} = a_{0,v-u \pmod n} = 1$ (prvi enačaj velja, ker je ničta vrstica u-kratni ciklični pomik u-te vrstice v levo). Po definiciji ničte vrstice potem velja $v - u = h_i \pmod n$ za nek $i \in I^+$, torej je $v = u + h_i \pmod n$ oziroma $u \sim v$. □

Dokaz trditve 3.2-IV (za neusmerjen primer).
KGA $\Rightarrow$ KGB. Iz $u \sim v$ sledi, da obstaja tak $s \in S$, da je $v = u + s \pmod n$, od tod pa, da obstaja tak $i \in I$, da je $v = u + \text{sign}(i)h_{|i|} \pmod n$. Če $i \in I^+$, dokaz nadaljujemo kot pri usmerjenem primeru, če pa je $i \in I^-$, je par (u, v) oblike $(u, u - h_{|i|} \pmod n) = (w + h_j \pmod n, w)$ pri $j = |i|$ in $w = u - h_j \pmod n$. Ker imamo pri neusmerjenem primeru opravka z neurejenimi pari, je par (u, v) enak paru $(w, w + h_j \pmod n)$, za nek $w \in V_n$ in $j \in I^+$, torej je v množici E. □

KGB $\Rightarrow$ KGC. Naj par (u, v) predstavlja par povezanih točk po definiciji KGB. Potem po lemi 3.2-I obstaja tak $i \in I^+$, da je (u, v) enak enemu izmed parov $(u, u + h_i \pmod n)$ in $(u, u - h_i \pmod n)$. V prvem primeru je torej (u, v) v množici E_i, v drugem pa v množici E_{-i}. V obeh primerih je torej $(u, v) \cup_{i\in I} E_i$. □

KGC $\Rightarrow$ KGD. Vzemimo par po definiciji KGC povezanih točk (u, v). Ker je $(u, v) \in \cup_{i\in I} E_i$, obstaja tak $i \in I$, da je $(u, v) \in E_i$. Par (u, v) se zato lahko zapiše v obliki $(u, v) = (u, u + \text{sign}(i) * h_{|i|} \pmod n)$. Ker po definiciji matrike A za vsak $i \in I$ velja $a_{0,h_{|i|}} = a_{0,-h_{|i|}} \pmod n$ in ker je u-ta vrstica u-kratni ciklični pomik ničte vrstice, je $a_{u,u+\text{sign}(i)*h_{|i|}} = 1$. Torej je par $(u, v) = (u, u + \text{sign}(i) * h_{|i|} \pmod n)$ povezan tudi po definiciji KGD. □

KGD $\Rightarrow$ KGA. Če iz točke u po definiciji KGD vodi povezava v točko v, potem velja $a_{u,v} = a_{0,v-u \pmod n} = 1$ (prvi enačaj velja, ker je ničta vrstica u-kratni ciklični pomik u-te vrstice v levo). Po definiciji ničte vrstice potem velja bodisi $v - u = h_i \pmod n$ bodisi $v - u = -h_i \pmod n$ za nek $i \in I^+$. Torej je $v = u + \text{sign}(i)h_{|i|} \pmod n$ za nek $i \in I$, oziroma $u \sim v$. □

Trditev 3.2-IV je s tem v celoti dokazana. Omenimo še, da bomo v nadaljevanju namesto izraza 2-krožen graf pogosto uporabljali izraz *dvokrožen graf*.

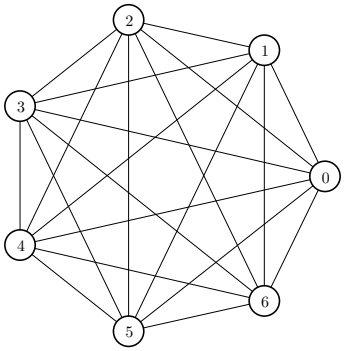
3.3 Uporaba krožnih grafov in primeri

Od dobre topologije omrežja pričakujemo, da bo imela majhen premer, majhno povprečno razdaljo med točkami, majhno stopnjo posamezne točke, majhno število povezav in da bo poleg tega čimbolj zanesljiva, razširljiva in simetrična [4]. Ker se nekatere od omenjenih lastnosti med seboj izključujejo, idealne topologije ni – topologija, ki je dobra v neki lastnosti, ima hibo v drugi. Obroč z usmerjenimi povezavami, na primer, ki ima preprosto strukturo in ga je zato lahko implementirati, ima velik premer in je zelo nezanesljiv, saj graf postane nepovezan že ob izpadu ene povezave ali ene točke. Popolnoma nasprotne lastnosti ima polno povezan graf, ki je ob majhnem premeru zelo zanesljiv, ima pa zato veliko število povezav in veliko stopnjo posamezne točke. Implementacija polno povezane topologije zahteva zapleteno strojno opremo, še posebej pri omrežjih z velikim številom vozlišč, zato se omrežja s to topologijo v praksi redko pojavljajo. Vmesno možnost med omenjenima ekstremoma predstavlja topologija krožnih grafov. Poleg sprejemljivo majhnega premera – ta je ob primernem izboru dolžine povezav že v primeru, ko je stopnja posameznega vozlišča 4, reda velikosti $\mathcal{O}(\sqrt{n})$ – so krožni grafi zanesljivi (povezanost k-krožnega grafa je k), dobro razširljivi in simetrični. Zaradi omenjenih lepih lastnosti se krožni grafi pogosto uporabljajo kot topologija za lokalna omrežja [1, 4, 7, 31, 32, 36, 39], komunikacijska omrežja [1, 6], ter kot povezovalna shema mreže procesorskih enot večprocesorskih sistemov [4, 6, 31, 36]. Med drugim so bili krožni grafi uporabljeni v računalniku ILLIAC IV, dandanes pa se uporabljajo v FDDI-token, SILK in SONET obročih [4, 39] ter v raznih paralelnih procesorskih sistemih in projektih (Intel Paragon, Cray T3D, MPP, MICROS) [46]. Topološke lastnosti [1, 2, 4, 7, 8, 10, 11, 15, 23, 25, 17, 28, 32, 37, 39, 40, 38, 45, 48, 51, 54] in lastnosti, povezane z usmerjenjem podatkov v krožnih grafih [2, 3, 6, 9, 12, 20, 21, 22, 24, 26, 27, 16, 30, 31, 33, 36, 41, 50, 49, 55], avtorji raziskujejo že desetletja, kljub temu pa v povezavi z njimi še vedno ostajajo odprta in nerešena vprašanja. Eno takih je, na primer, vprašanje časovne zahtevnosti izračuna premera pri $k > 2$ [54].

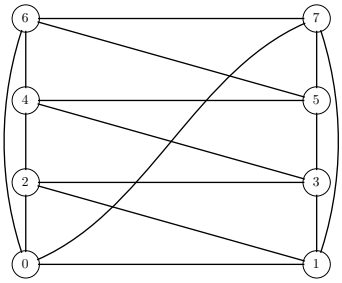
Primeri krožnih grafov

Krožni grafi so velika družina grafov, med katere sodijo tudi cikli, polni grafi ter nekateri torusi. Najpreprostejši primeri dvokrožnih grafov so prikazani na slikah 3.1-I na strani 20 ter 3.2-I na strani 23, slika 3.3-I pa prikazuje: (a) 3-krožen graf $G(7;\ 1, 2, 3)$, ki predstavlja polni graf na sedmih točkah, (b) nestandardno razporeditev točk dvokrožnega grafa $G(8;\ 1, 2)$, (c) zaviti torus $G(16;\ 1, 4)$ – podoben, 4-krat večji graf $G(64;\ 1, 8)$, je bil uporabljen v računalniku ILLIAC IV – ter (d) rekurzivni cirkulant $G(36;\ 1, 6)$; rekurzivni cirkulanti so podrobneje predstavljeni v [33].

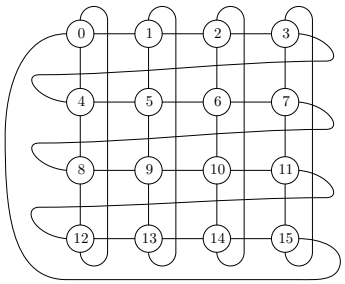
V literaturi avtorji večkrat omenjajo tri posebne primere družin krožnih grafov. Natančneje so te družine opisane v [2], tu jih omenimo le na kratko. Prva družina DDLCN grafov (*angl. Distributed Double Loop Computer Network*) so grafi, ki jih dobimo z razširitvijo usmerjenega obroča – vse povezave obroča podvojimo in duplikate usmerimo v obratno smer, kot so usmerjeni originali. Tako se poveča zanesljivost omrežja, saj se povezanost grafa poveča iz 1 na 2, za polovico pa se zmanjša tudi premer. Z oznakami



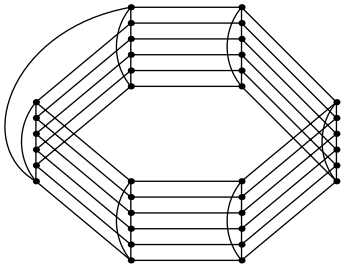
(a) Polno povezan graf
$G(7;\ 1, 2, 3)$



(b) dvokrožen graf
$G(8;\ 1, 2)$



(c) Zaviti torus
$G(16;\ 1, 4)$



(d) Rekurzivni cirkulant
$G(36;\ 1, 6)$

Slika 3.3-I: Primer nekaterih krožnih grafov

prejšnjega razdelka lahko, čeprav malo nenatančno, grafe družine DDLCN označimo z $G(n;\ -1, +1)$. S povečanjem dolžin povezav enega od obročev dobimo družini DCLN (*angl. Daisy Chain Loop Networks*) in FLBHN (*angl. Forward Loop Backward Hop Networks*). Prva predstavlja grafe $G(n;\ -1, +2)$, druga pa grafe $G(n;\ -1, +h)$ pri poljubnem $h > 0$. Premer grafov družine FLBHN je $\lfloor \frac{n}{h+1} \rfloor + (h-1)$ in doseže svoj minimum pri $h = \lfloor \sqrt{n} \rfloor$.

3.4 Lastnosti krožnih grafov

3.4.1 Predstavitev

Krožni grafi se od drugih grafov razlikujejo tudi po številu podatkov, ki so potrebni za njihovo predstavitev. Splošen graf lahko predstavimo tako, da podamo množico točk V in množico povezav E; če je moč množice V enaka n, je moč množice E reda velikosti $\mathcal{O}(n^2)$, tak je tudi red velikosti podatkov, ki so potrebni za predstavitev splošnega grafa. Mnogo manj podatkov pa potrebujemo za predstavitev k-krožnega grafa; graf $G(n;\ \pm h_1, \ldots, \pm h_k)$ je namreč natančno podan s števili $n, h_1, h_2, \ldots, h_k$. Ker za pred-

stavitev poljubnega števila, ki je manjše ali enako n, zadostuje $\log_2 n$ bitov, je pri fiksnem k red velikosti podatkov, ki so potrebni za predstavitev krožnega grafa, enak $\mathcal{O}(\log n)$.

Kompakten način predstavitve krožnih grafov se odraža tudi pri ocenjevanju zahtevnosti algoritmov. Namreč, algoritem, ki je polinomski v n, je eksponenten v velikosti opisa krožnih grafov. Tako je, na primer, klasičen $\mathcal{O}(n^2)$ algoritem za izračun premera splošnega grafa eksponenten glede na velikost predstavitve krožnega grafa. Če iščemo polinomski algoritem za izračun premera krožnih grafov, mora biti njegova časovna zahtevnost reda velikosti $\mathcal{O}(\log^p n)$, za nek $p \in \mathbb{N}$.

3.4.2 Simetrija

Pomembna lastnost krožnih grafov je njihova vozliščna simetrija, ki jo formalno definiramo s pomočjo avtomorfizma grafov [47].

Definicija 3.4-I. *Naj bo $\mathcal{A}$ bijektivna preslikava na končni množici vozlišč V grafa $G(V, E)$. Preslikava $\mathcal{A}$ je* avtomorfizem *grafa $G(V, E)$, če za poljuben par točk $u, v \in V$, za katerega je $(u, v) \in E$, velja $(\mathcal{A}(u), \mathcal{A}(v)) \in E$.*

Avtomorfizem grafa $G(V, E)$ je torej vsaka bijektivna preslikava (permutacija) na množici V, ki ohranja sosednost.

> Primer 3.4-I. Za vsak $i \in \{0, 1, \ldots, n-1\}$ je preslikava $\mathcal{A}_i : V \to V$ definirana z $\mathcal{A}_i(v) = v + i \pmod n$ avtomorfizem krožnega grafa $G = G(n;\ h_1, \ldots, h_k)$. Res, če sta namreč točki u in v v G sosednji, sta v G sosednji tudi točki $u + i \pmod n$ in $v+i \pmod n$, saj iz $u = v+h \pmod n$ neposredno sledi $u+i = v+i+h \pmod n$.

Definicija 3.4-II. *Graf $G = G(V, E)$ je* vozliščno simetričen *(*angl. vertex-transitive*), če za vsak par točk $u, v \in V$ obstaja tak avtomorfizem grafa G, ki točko u preslika v točko v.*

Trditev 3.4-III. *Krožni grafi so vozliščno simetrični.*

Dokaz. Naj bosta u in v poljubni točki danega krožnega grafa $G(n;\ h_1, \ldots, h_k)$. Če z w označimo $w = v - u \pmod n$, potem je preslikava $\mathcal{A}_w : x \mapsto x + w \pmod n$ avtomorfizem (glej primer 3.4-I), in velja $\mathcal{A}_w(u) = v$. □

Pri razvoju in analizi algoritmov za usmerjanje podatkov smo poleg dejstva, da lahko v procesu usmerjanja vse točke igrajo enako vlogo (t.j. v vseh lahko teče enak usmerjevalni algoritem), pogosto uporabili naslednjo preprosto vendar zelo uporabno posledico vozliščne simetrije krožnih grafov.

Posledica 3.4-IV. *Naj bosta u in v poljubni točki danega krožnega grafa in naj bo $e = \mathcal{E}_{i_1}\,\mathcal{E}_{i_2}\,\cdots\,\mathcal{E}_{i_l}$ zaporedje tipov povezav, ki povezujejo točki u in v. Potem e povezuje tudi točki 0 in $w = v - u \pmod n$ tega grafa.*

Dokaz. Avtomorfizem $\mathcal{A} : x \mapsto x - u \pmod n$ vozlišče u preslika v 0, vozlišče v pa v w, zato zaporedje tipov povezav, ki v originalnem grafu povezuje točki u in v, v sliki $\mathcal{A}(G(V,E))$ povezuje točki 0 in w. □

Po tej posledici se problem iskanja najkrajših poti med vsemi pari točk krožnega grafa poenostavi na problem iskanja najkrajših poti med točko 0 in drugimi točkami.

Podobno kot smo definirali vozliščno, lahko definiramo tudi ločno simetrijo grafa in sicer takole: graf G je ločno simetričen (*angl. arc-transitive*), če za vsak par parov sosednjih točk (u,v) in (x,y) obstaja avtomorfizem, ki u preslika v x in v v y. Čeprav obstajajo krožni grafi, ki so ločno simetrični (na primer graf $G(8;\ \pm 1, \pm 3)$), v splošnem to ne velja.

Trditev 3.4-V. *Krožni grafi v splošnem niso ločno simetrični.*

Kot dokaz te trditve si oglejmo graf $G(8;\ \pm 1, \pm 2)$. Povezava, ki v tem grafu povezuje točki 1 in 2, leži na ciklu dolžine 8, povezava, ki povezuje točki 1 in 3, pa na ciklu dolžine 4. Avtomorfizmi grafa G torej prve povezave v nobenem primeru ne morejo preslikati v drugo, zato G ni ločno simetričen.

3.4.3 Povezanost

Krožen graf, ki je podan s parametri $n, h_1, \ldots, h_k$, je povezan tedaj in le tedaj, ko števila $n, h_1, \ldots, h_k$ nimajo skupnega faktorja, oziroma ko velja,

$$\gcd(n, h_1, h_2, \ldots, h_k) = 1.$$

To velja tako za usmerjene kot tudi za neusmerjene krožne grafe. Res, če je namreč $g = \gcd(n, h_1, h_2, \ldots, h_k) \neq 1$, potem lahko iz točke 0 dosežemo le tiste točke grafa, katerih oznaka je večkratnik števila g. V nasprotnem primeru pa, ko je $g = 1$, obstajajo taka števila $a_0', a_1', \ldots, a_k'$, da velja $a_0' n + a_1' h_1 + a_2' h_2 + \cdots + a_k' h_k = 1$ (ta števila lahko izračunamo z razširjenim Evklidovim algoritmom [35, 42]). Če za vsak $i = 1, \ldots, n$ z a_i označimo ostanek števila a_i' pri deljenju z n, to je $a_i = a_i' \bmod n$, potem so a_i naravna števila in velja

$$a_1 h_1 + a_2 h_2 + \cdots + a_k h_k \equiv 1 \pmod n.$$

Od tod pa sledi, da je točka 0 povezana s točko 1 in (posledično) z vsemi drugimi točkami grafa [32].

Ker je $\gcd(n, h_1, h_2, \ldots, h_k) = 1$ pogoj za povezanost grafa, bomo ta pogoj v nadaljevanju dela implicitno predpostavljali. Preprosto se prepričamo, da je vsak povezan krožen graf po povezavah in po točkah k-povezan [4] (t.j. graf ostane povezan, če odstranimo manj kot k povezav oziroma točk [56]), z malo več truda pa, da je vsak k-krožen graf po povezavah $2k$-povezan in celo, da je vsak dvokrožen graf z več kot 5 točkami po točkah 4-povezan [40].

3.4.4 Premer

Premer krožnih grafov je odvisen od števil n in $h_1, \ldots, h_k$ in je reda velikosti $\mathcal{O}(n)$. Poleg krožnih grafov, pri katerih je premer približno enak n, obstajajo tudi primeri z bistveno manjšim premerom. Natančne vrednosti za premer krožnega grafa ne poznamo niti za dvokrožne niti za splošne k-krožne grafe, poznamo pa ocene za njegovo spodnjo mejo $lb(n)$. Te ocene so odvisne od usmerjenosti povezav in vrednosti števila k, v vseh primerih pa so reda velikosti $\Theta(n^{1/k})$. Obstaja neskončna družina dvokrožnih grafov, katerih premer je enak $lb(n)$, za splošne k-krožne pa je velikost te družine še odprto vprašanje [4]. Oglejmo si vrednosti $lb(n)$ za posamezen primer.

Usmerjeni dvokrožni grafi. Spodnja meja za premer usmerjenega dvokrožnega grafa je $lb(n) = \lceil\sqrt{3n}\rceil - 2$. V [32] je predstavljena neskončna družina grafov, katerih premer je enak $lb(n)$; obstaja tudi karakterizacija števil n, pri katerih taki grafi obstajajo [4].

Neusmejeni dvokrožni grafi. Spodnja meja za premer neusmerjenega dvokrožnega grafa je $lb(n) = \lceil(-1 + \sqrt{2n-1})/2\rceil$. To je natančna spodnja meja, saj velja

$$\text{diam}(G(n; \pm lb(n), \pm(lb(n)+1))) = lb(n);\ \forall n\ [40].$$

Poleg te obstajajo tudi druge družine dvokrožnih grafov, katerih premer je enak $lb(n)$. Če pri danem n za noben h ne velja diam($G(n;\ \pm 1, \pm h)$)=$lb(n)$, potem pravimo, da je n suboptimalno število. Množica suboptimalnih števil vsebuje neskončno mnogo elementov, karakterizacija suboptimalnih števil pa ni znana [54].

Neusmerjeni k-krožni grafi. Spodnja meja za premer je $(k!n)^{1/k} - \frac{1}{2}(k+1)$, ne vemo pa, ali obstaja neskončna družina grafov, ki ta premer dosežejo [4]. Dokažemo pa lahko, da obstaja neskončno števil n, pri katerih obstajajo taka števila $h_2, h_3, \ldots, h_k$, da je premer grafa $G(n;\ 1, h_2, \ldots, h_k)$ manjši ali kvečjemu enak $(((k-1)!c)^{1/(k-1)} + 1/c)n^{1/k} - (k+2)/2$, pri poljubno izbranem $c \in \mathbb{R} - \{0\}$ [40].

Za računanje premera krožnih grafov lahko uporabimo standardne polinomske algoritme za računanje premera v splošnih grafih, toda ti imajo za krožne grafe eksponentno časovno zahtevnost. Za splošen primer (to je pri poljubnem n in poljubnem, nefiksnem k) je računanje premera krožnih grafov NP-težek problem [6]. Pri fiksnem $k = 2$ je v [54]

predstavljen $\mathcal{O}(\log n)$ algoritem za računanje premera, kompleksnost tega problema pri fiksnem $k > 3$ pa je še odprto vprašanje [54].

3.5 Sprehodi in poti v krožnih grafih

Sprehod v krožnih grafih je definiran kot poljubno zaporedje povezav oblike

$$u_0u_1, u_1u_2, u_2u_3, \ldots, u_{l-1}u_l \tag{3.1}$$

pri $u_i \in V_n$ za vsak $i \in \{0, 1, \ldots, l-1, l\}$ in $(u_i, u_{i+1}) \in E$ za vsak $i \in \{0, 1, \ldots, l-2, l-1\}$. Pogosto za zapis sprehoda uporabimo bolj jedrnat zapis in namesto 3.1 pišemo $u_0\ u_1\ u_2\ \ldots\ u_l$. Točko u_0 imenujemo **začetna**, u_l pa **končna** točka sprehoda. Dolžina sprehoda je enaka l, torej številu povezav, ki sprehod sestavljajo [56].

> Primer 3.5-I. Sprehod $s = (7,0)(0,3)(3,6)(6,8)$ med točkama 7 in 8 dvokrožnega grafa $G(9;\ \pm 2, \pm 3)$ lahko zapišemo tudi kot zaporedje $s = 7\ 0\ 3\ 6\ 8$. Dolžina sprehoda s je 4.

3.5.1 Vrstni red povezav vzdolž sprehoda

Sprehod s iz primera 3.5-I sestavljajo povezava tipa $\mathcal{E}_1$ ($u \mapsto u+2 \pmod 9$), dve povezavi tipa $\mathcal{E}_2$ ($u \mapsto u+3 \pmod 9$) ter ponovno povezava tipa $\mathcal{E}_1$:

$$s : 7 \xrightarrow{\mathcal{E}_1} 0 \xrightarrow{\mathcal{E}_2} 3 \xrightarrow{\mathcal{E}_2} 6 \xrightarrow{\mathcal{E}_1} 8.$$

V končno točko 8 smo s sprehodom s prišli tako, da smo začetni točki 7 prišteli najprej 2, potem dvakrat 3 in na koncu ponovno 2. Pri tem smo uporabili seštevanje po modulu 9.

$$\underbrace{8}_{\substack{\text{končna}\\ \text{točka}}} = \underbrace{7}_{\substack{\text{začetna}\\ \text{točka}}} + \underbrace{2}_{\substack{\text{povezava}\\ \text{tipa } \mathcal{E}_1}} + \underbrace{3}_{\substack{\text{povezava}\\ \text{tipa } \mathcal{E}_2}} + \underbrace{3}_{\substack{\text{povezava}\\ \text{tipa } \mathcal{E}_2}} + \underbrace{2}_{\substack{\text{povezava}\\ \text{tipa } \mathcal{E}_1}} \pmod 9 \tag{3.2}$$

Ker je seštevanje komutativna operacija, bi dobili isti končni rezultat 8, če bi pri dani začetni točki 7 uporabili povezave tipa $\mathcal{E}_1$, $\mathcal{E}_2$, $\mathcal{E}_2$, $\mathcal{E}_1$ v poljubnem vrstnem redu. To ugotovitev lahko posplošimo in jo zapišemo v obliki naslednje leme.

Trditev 3.5-I. *Končna točka sprehoda v krožnih grafih je odvisna le od začetne točke in od števila povezav posameznega tipa, ki sprehod sestavljajo.*

Trditev 3.5-I, ki poenostavljeno pravi, da "vrstni red povezav vzdolž sprehoda za končni cilj ni pomemben" [6, 54], bo za nas zanimiva pri reševanju usmerjevalnih problemov v krožnih grafih, pri katerih nas pogosto zanima le "učinek" danega sprehoda (t.j. cilj, do katerega nas ta pripelje). Možnost spreminjanja vrstnega reda povezav bomo s pridom uporabili za preprečevanje in/ali reševanje usmerjevalnih konfliktov.

3.5.2 Regularni sprehodi

Sprehod, ki povezuje dve točki danega grafa in vsebuje par nasprotnih povezav, to je povezav tipa $\mathcal{E}_i$ in $\mathcal{E}_{-i}$ za nek $i \in I$ (glej primer 3.5-II), je "potraten", saj se učinek omenjenih povezav vzdolž sprehoda izniči. Sprehod, ki ga dobimo z izločitvijo nasprotnih povezav, dano začetno točko povezuje z isto končno točko. Da je to res, se zlahka prepričamo, če sprehod predstavimo v obliki enačbe, podobno kot smo to storili v 3.2. Ena od povezav $\mathcal{E}_i$ in $\mathcal{E}_{-i}$ v taki enačbi prispeva vrednost h_i, druga pa $-h_i$, skupaj torej 0.

> Primer 3.5-II. Naj bo sprehod s_1 od točke 7 do točke 8 v grafu $G(9;\ \pm 2, \pm 3)$ sestavljen iz dveh povezav tipa $\mathcal{E}_1$, ene povezave tipa $\mathcal{E}_2$ in dveh povezav tipa $\mathcal{E}_{-2}$. Da ta sprehod res povezuje točki 7 in 8, se prepričamo z naslednjo enačbo: $7 + 2 + 2 + 3 + (-3) + (-3) = 7 + 2 + 2 - 3 = 8 \pmod 9$. Iz te enačbe vidimo, da se je učinek povezav "nasprotnega" tipa ($\mathcal{E}_2$ in $\mathcal{E}_{-2}$) izničil, oziroma da ni vplival na cilj sprehoda (sprehod bi pripeljal v isti cilj tudi v primeru, če bi obe povezavi preprosto izpustili). Sprehod s_2, ki je sestavljen iz dveh povezav tipa $\mathcal{E}_1$ in ene $\mathcal{E}_{-2}$, pripelje v isti cilj kot s_1 in je za dve povezavi krajši od s_1. Sprehod s_1 je potraten.

Sprehod, ki vsebuje par nasprotnih povezav, ni najkrajši, saj je od njega za dva koraka krajši sprehod z odstranjenima odvečnima povezavama. Kratke sprehode bomo torej iskali le med tistimi, ki ne vsebujejo povezav nasprotnih tipov. Imenovali jih bomo *regularni* sprehodi [41].

Definicija 3.5-II. *Sprehod v k-krožnem grafu je* regularen, *če za noben $i \in I$ ne vsebuje hkrati povezav tipa $\mathcal{E}_i$ in $\mathcal{E}_{-i}$.*

Predstavitev sprehoda s številom povezav posameznega tipa
Ena od posledic trditve 3.5-I je, da lahko sprehod v krožnih grafih, kadar nas zanima le njegov učinek (cilj), predstavimo namesto z zaporedjem povezav s številom povezav posameznega tipa. V neusmerjenem k-krožnem grafu je $2k$ različnih tipov povezav, torej je za predstavitev poljubnega sprehoda potrebnih le $2k$ naravnih števil.

Na še bolj jedrnat način pa lahko predstavimo regularen sprehod, saj ta vsebuje največ k različnih tipov povezav. Ker regularen sprehod ne more hkrati vsebovati povezave tipa $\mathcal{E}_i$ in $\mathcal{E}_{-i}$, lahko za vsak $i \in I$ število povezav omenjenih tipov predstavimo samo z enim številom. Predstavitev regularnega sprehoda tako podamo s k-terico celih števil $(x_1, x_2, \ldots, x_k)$, pri čemer x_i nosi informacijo o številu povezav tipa $\mathcal{E}_i$ in $\mathcal{E}_{-i}$ [6, 54]. Natančneje, če je regularen sprehod s predstavljen s k-terico $(x_1, x_2, \ldots, x_k)$, potem za vsak $i \in I^+$ velja:

- če je $x_i > 0$, potem s vsebuje x_i povezav tipa $\mathcal{E}_i$,
- če je $x_i < 0$, potem s vsebuje $|x_i|$ povezav tipa $\mathcal{E}_{-i}$,
- če je $x_i = 0$, potem s ne vsebuje niti povezav tipa $\mathcal{E}_i$ niti $\mathcal{E}_{-i}$.

k-terica števil $(x_1, x_2, \ldots, x_k)$ predstavlja vse sprehode, ki za vsak $i \in I^+$, pri katerem je $x_i \neq 0$, vsebujejo $|x_i|$ povezav tipa $\mathcal{E}_{i*}$ sign(x_i).

3.5.3 Ekvivalentni sprehodi

Nad množico vseh regularnih sprehodov v danem grafu vpeljimo tako relacijo, da bo povezovala tiste sprehode, ki se med seboj razlikujejo le po vrstnem redu tipov povezav.

Definicija 3.5-III. *Naj bo S množica vseh regularnih sprehodov v krožnem grafu $G(n;\ \pm h_1, \ldots, \pm h_k)$. Sprehoda $s_1, s_2 \in S$ sta v relaciji R ($s_1 R s_2$), če vsebujeta isto število povezav posameznega tipa.*

Relacija R je ekvivalenčna relacija, o čemer se prepričamo v naslednji trditvi.

Trditev 3.5-IV. *Relacija R nad množico vseh regularnih sprehodov krožnega grafa $G(n;\ \pm h_1, \ldots, \pm h_k)$ je ekvivalenčna relacija.*

Dokaz. Naj bo regularen sprehod s_1 predstavljen s k-terico števil $(x_1, x_2, \ldots, x_k)$, regularen sprehod s_2 pa s k-terico $(y_1, y_2, \ldots, y_k)$. Potem velja

$$s_1 R s_2 \iff (x_1, x_2, \ldots, x_k) = (y_1, y_2, \ldots, y_k).$$

Ker je relacija "=" ekvivalenčna relacija nad množico $\mathbb{Z}^k$, je tudi relacija R nad množico vseh regularnih sprehodov S ekvivalenčna. □

Ekvivalenčni razred je natančno določen s k-terico števil $(x_1, x_2, \ldots, x_k)$, pri čemer x_i določa število povezav tipa $\mathcal{E}_i$ in $\mathcal{E}_{-i}$.

Definicija 3.5-V. *Ekvivalenčni razred relacije R nad množico vseh regularnih sprehodov krožnega grafa $G(n;\ \pm h_1, \ldots, \pm h_k)$, ki vsebuje sprehode s po $|x_i|$ povezav tipa $\mathcal{E}_{\hat{\imath}}$ (pri $\hat{\imath} =$ sign(x_i) $* i$), za vsak $i \in I^+$, bomo označili z $[x_1, x_2, \ldots, x_k]$.*

Dolžina sprehodov ekvivalenčnega razreda $[x_1, x_2, \ldots, x_k]$ je $\sum_{i \in I^+} |x_i|$.

Pomen izraza *sprehod* $(x_1, x_2, \ldots, x_k)$
V nadaljevanju tega dela nas pogosto ne bo zanimal konkreten sprehod med dvema točkama, ampak bomo iskali le ekvivalenčni razred sprehodov z določenimi lastnostmi. Zaradi preprostosti zapisa bomo uporabljali izraz *sprehod* $(x_1, x_2, \ldots, x_k)$, ki bo označeval enega od sprehodov ekvivalenčnega razreda $[x_1, x_2, \ldots, x_k]$.

3.5.4 Iskanje sprehoda in kongruenčna enačba

Vrnimo se še za hip k sprehodu $s = (7,0)(0,3)(3,6)(6,8)$ iz primera 3.5-I. Pomen urejenega para (2,2), ki opisuje sprehod s med točkama 7 in 8 v grafu $G(9;\ \pm 2, \pm 3)$, razberemo iz enačbe (3.2), ki se (malo preurejena) glasi: $7+\mathbf{2} * 2+\mathbf{2} * 3 = 8 \pmod 9$, oziroma z besedami: sprehod z začetkom v točki 7, ki vsebuje dve povezavi dolžine 2 in dve povezavi dolžine 3, se konča v točki 8. Podobno enačbo lahko napišemo tudi za splošen primer. Namreč, k-terica $(x_1, x_2, \ldots, x_k)$ določa ekvivalenčni razred sprehodov med točkama u in v krožnega grafa $G(n;\ \pm h_1, \ldots, \pm h_k)$ natanko tedaj, ko velja

$$u + x_1 h_1 + x_2 h_2 + \ldots + x_k h_k \equiv v \pmod n. \tag{3.3}$$

Iskanje ekvivalenčnega razreda sprehodov med danima točkama u in v krožnega grafa je torej ekvivalentno iskanju rešitve kongruenčne enačbe (3.3). Ta enačba ima rešitev (pri poljubnih u in v) natanko tedaj, ko velja $\gcd(n, h_1, \ldots, h_k) = 1$; če je pogoj izpolnjen, je rešitev neskončno mnogo. Natančen postopek reševanja enačbe (3.3) bomo opisali v razdelku 3.8; videli bomo, da pri reševanju uporabimo Evklidov algoritem, ki rešitev poišče v $\mathcal{O}(\log n)$ časa. Pogoj, da števila n in $h_i, i \in I^+$ ne smejo imeti skupnega faktorja, je hkrati tudi pogoj za povezanost krožnega grafa, kar seveda ni presenetljivo. Graf je namreč povezan, kadar obstaja sprehod med poljubnim parom njegovih točk, torej natanko tedaj, ko obstaja rešitev enačbe (3.3).

Iz enačbe (3.3) je razvidno, da je vsak sprehod med točkama u in v hkrati tudi sprehod med točkama 0 in $w = v - u \pmod n$, saj je vsaka k-terica $(x_1, x_2, \ldots, x_k)$, ki je rešitev enačbe (3.3), hkrati tudi rešitev enačbe $0 + x_1 h_1 + x_2 h_2 + \ldots + x_k h_k = v - u \pmod n$. Ob upoštevanju te trditve se problem iskanja sprehodov med vsemi pari točk krožnega grafa prevede na problem iskanja sprehoda med točko 0 in drugimi točkami grafa (glej tudi posledico 3.4-IV).

3.5.5 Pot in najkrajša pot v krožnih grafih

Pot v krožnem grafu je definirana kot sprehod $u_0\ u_1\ u_2\ \ldots\ u_l$, v katerem se nobena izmed točk ne pojavi več kot enkrat [56]. Med vsemi potmi, ki povezujejo dve točki danega krožnega grafa, so še posebej pomembne tako imenovane *najkrajše poti*, oziroma poti, ki vsebujejo minimalno število povezav.

Definicija 3.5-VI. Najkrajša pot *med točkama u in v krožnega grafa je vsaka pot med u in v najkrajše dolžine.*

Med dvema točkama krožnega grafa lahko obstaja več najkrajših poti. Če je namreč neka pot, ki vsebuje $|x_i|$ povezav tipa $\mathcal{E}_{i*}$ sign(x_i) $(\forall i \in I^+)$ najkrajša pot med u in v, potem je vsaka pot ekvivalenčnega razreda $[x_1, x_2, \ldots, x_k]$ najkrajša. Prepričajmo se v to še formalno.

Trditev 3.5-VII. *Naj bosta u in v poljubni točki k-krožnega grafa in naj bo k-terica celih števil $(x_1, x_2, \ldots, x_k)$ taka rešitev enačbe (3.3), da za vsako drugo rešitev $(y_1, y_2, \ldots, y_k)$ te enačbe velja $\sum_{i\in I^+} |x_i| \leq \sum_{i\in I^+} |y_i|$. Potem je vsak sprehod iz ekvivalenčnega razreda $[x_1, x_2, \ldots, x_k]$ pot.*

Dokaz. Pa recimo, da obstaja tak sprehod $s = u_0\ u_1\ u_2\ \ldots\ u_l$ ekvivalenčnega razreda $[x_1, x_2, \ldots, x_k]$, ki ni pot. Naj bo u_i tista točka sprehoda s, ki se ponovi, in naj bo s' podsprehod sprehoda s, ki se začne in konča v točki u_i. Za vse elemente k-terice $(z_1, z_2, \ldots, z_k)$, ki predstavlja podsprehod s', velja $|z_i| \leq |x_i|$ in sign(z_i) = sign(x_i). Za vsak $i \in I^+$ je torej $|x_i - z_i| \leq |x_i|$; v primerih, ko $z_i \neq 0$, velja strogi neenačaj. Ker je s' sklenjen sprehod, velja $u_i + z_1h_1 + \ldots + z_kh_k = u_i \pmod n$, oziroma $z_1h_1 + \ldots + z_kh_k = 0 \pmod n$. Zato je $(x - z) = (x_1 - z_1, x_2 - z_2, \ldots, x_k - z_k)$ rešitev enačbe 3.3 in velja $\sum_{i\in I^+} |x_i - z_i| < \sum_{i\in I^+} |x_i|$, kar pa je v nasprotju s predpostavko. □

Posledica 3.5-VIII. *Naj bosta u in v dani točki k-krožnega grafa $G(n;\ \pm h_1, \ldots, \pm h_k)$ in naj bo k-terica celih števil $(x_1, x_2, \ldots, x_k)$ taka rešitev enačbe (3.3), da za vsako drugo rešitev $(y_1, y_2, \ldots, y_k)$ te enačbe velja $\sum_{i\in I^+} |x_i| \leq \sum_{i\in I^+} |y_i|$. Vsako pot ekvivalenčnega razreda $[x_1, x_2, \ldots, x_k]$ je najkrajša pot med točkama u in v.*

Iskanje najkrajše poti v krožnem grafu je ekvivalentno iskanju take rešitve $(x_1, x_2, \ldots, x_k)$ kongruenčne enačbe (3.3), ki med vsemi rešitvami te enačbe minimizira vsoto

$$|x_1| + |x_2| + \ldots + |x_k|. \tag{3.4}$$

Dejstvo, da obstaja več ekvivalentnih poti z isto dolžino, se da s pridom izkoristiti pri razvoju usmerjevalnih algoritmov (glej poglavje 4), kjer za prenos podatkov uporabimo najkrajšo pot med začetno in končno točko. Več ekvivalentnih najkrajših poti pomeni več možnosti za reševanje morebitnih težav, ki se pojavijo med usmerjanjem (trki na povezavah, izpad katere od povezav ...). Poleg tega včasih obstaja več kot en ekvivalenčni razred najkrajših poti med dvema točkama danega grafa.

Primer 3.5-III. V grafu $G(11;\ \pm 2, \pm 3)$ sta ekvivalenčna razreda, ki opisujeta najkrajše poti med točkama 0 in 6 enaka [0,2] in [-1,-1], saj je $0 \xrightarrow{+3} 3 \xrightarrow{+3} 6$ in $0 \xrightarrow{-2} 9 \xrightarrow{-3} 6$.

3.6 Dvokrožni grafi in mreža $\mathbb{Z}_2^2$

Med splošne principe reševanja težkih problemov sodi tudi metoda uvedbe novega okolja in prevedbe prvotnega problema v ekvivalenten problem, povezan s tem okoljem. Seveda pri tem predpostavljamo, da znamo ekvivalenten problem rešiti in da znamo njegovo rešitev prevesti v rešitev prvotnega problema. V našem primeru, to je pri reševanju problemov povezanih z dvokrožnimi grafi, bomo kot novo okolje uvedli celoštevilsko mrežo $\mathbb{Z}_2^2$,

v kateri znamo lepo definirati ekvivalentne probleme za probleme, kot sta iskanje premera in najkrajših poti med točkami, poleg tega pa znamo v njej te probleme tudi rešiti.

Definicija 3.6-I. Celoštevilska mreža $\mathbb{Z}_2^2$ *je množica celoštevilskih parov v ravnini,*

$$\mathbb{Z}_2^2 = \{(x_1, x_2);\ x_1, x_2 \in \mathbb{Z}\}.$$

Zvezo med celoštevilsko mrežo $\mathbb{Z}_2^2$ in izbranim dvokrožnim grafom $G(n;\ \pm h_1, \pm h_2)$ določa naslednje pravilo: element (x_1, x_2) mreže predstavlja ekvivalenčni razred $[x_1, x_2]$ sprehodov v grafu.

> Primer 3.6-I. Element $(4, -5)$ mreže $\mathbb{Z}_2^2$, ki pripada grafu $G(8;\ \pm 2, \pm 3)$, predstavlja tiste sprehode v grafu, ki iz začetne točke u pripeljejo v končno točko $u + \mathbf{4} * 2 + (\mathbf{-5}) * 3 = u - 7 = u + 1 \pmod 8$ in vsebujejo 4 povezave tipa $\mathcal{E}_1$ ter 5 povezav tipa $\mathcal{E}_{-2}$.

3.6.1 Označevalna funkcija

Dodatna vez med mrežo in grafom je tako imenovana **označevalna funkcija**. Ta elementu mreže, ki predstavlja ekvivalenčni razred sprehodov z začetno in končno točko 0 in w, priredi točko w v grafu. Označevalno funkcijo mreže, ki pripada grafu $G(n;\ \pm h_1, \pm h_2)$, označimo z l in podamo z

$$l : \mathbb{Z}_2^2 \to V_n;\ \ l((x_1, x_2)) = x_1 h_1 + x_2 h_2 \pmod n.$$

Ker je graf $G(n;\ \pm h_1, \pm h_2)$ po predpostavki povezan, je l surjektivna funkcija (vsaka točka $v \in V_n$ se kot oznaka elementov mreže pojavi vsaj enkrat). Hitro tudi vidimo, da se nobena oznaka ne pojavi samo enkrat. Namreč, če je w oznaka elementa (x_1, x_2), potem je tudi oznaka elementov $(x_1 + i * n, x_2)$ in $(x_1, x_2 + i * n)$, za vsak $i \in \mathbb{Z}$. To je res, saj iz $x_1 h_1 + x_2 h_2 = w \pmod n$ neposredno sledi $(x_1 + i * n)h_1 + x_2 h_2 = x_1 h_1 + (x_2 + i * n)h_2 = w \pmod n$. Iz pravkar povedanega pa sledi, da se vsaka oznaka $w \in V_n$ v mreži pojavi neskončno mnogokrat. Slika 3.6-I(a) prikazuje označeno mrežo, ki pripada grafu $G(9;\ \pm 2, \pm 3)$.

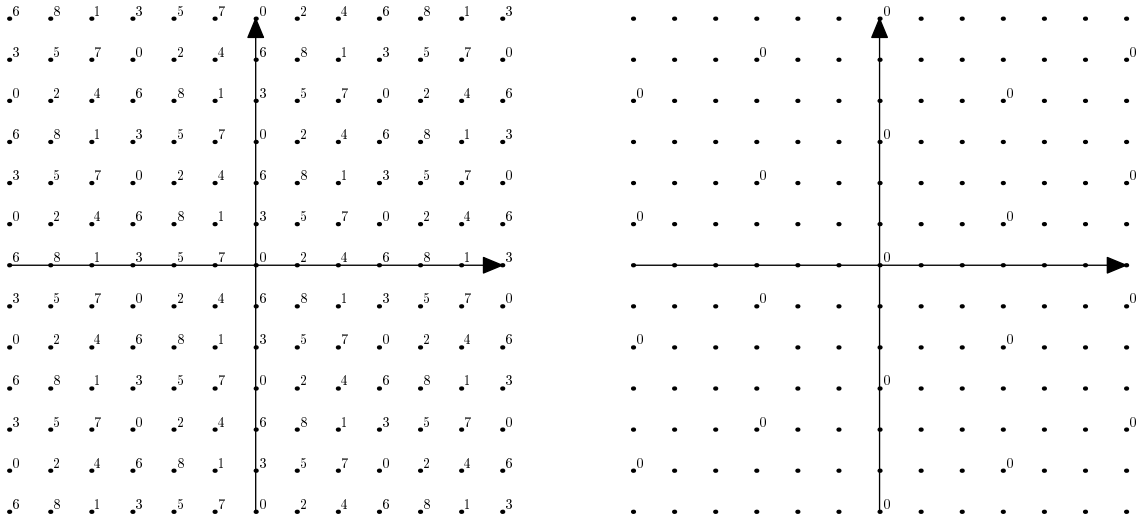
Podmnožice elementov z isto oznako

Mreža $\mathbb{Z}_2^2$ razpade na podmnožice, v katerih so elementi z isto oznako. Množico elementov z oznako w bomo označili z X_w,

$$X_w = \{x \in \mathbb{Z}_2^2;\ l(x) = w\}.$$

Videli smo že, da nobena od množic X_w ni prazna in da vse vsebujejo neskončno mnogo elementov.

Med vsemi množicami X_w je posebej pomembna množica X_0. Ker bomo njej posvetili več pozornosti kot drugim, je upravičeno tudi dodatno poimenovanje. Množici X_0 bomo preprosto rekli kar množica X,



Slika 3.6-I: (a) Označena mreža $\mathbb{Z}_2^2$ in (b) množica X, oboje za graf $G(9;\ \pm 2, \pm 3)$

$$X := X_0.$$

Elementi množice X pripadajo ekvivalenčnim razredom obhodov, to je sprehodov, ki se začnejo in končajo v isti točki. Slika 3.6-I(b) prikazuje mrežo $\mathbb{Z}_2^2$ za graf $G(9;\ \pm 2, \pm 3)$, v kateri so označeni le elementi množice X.

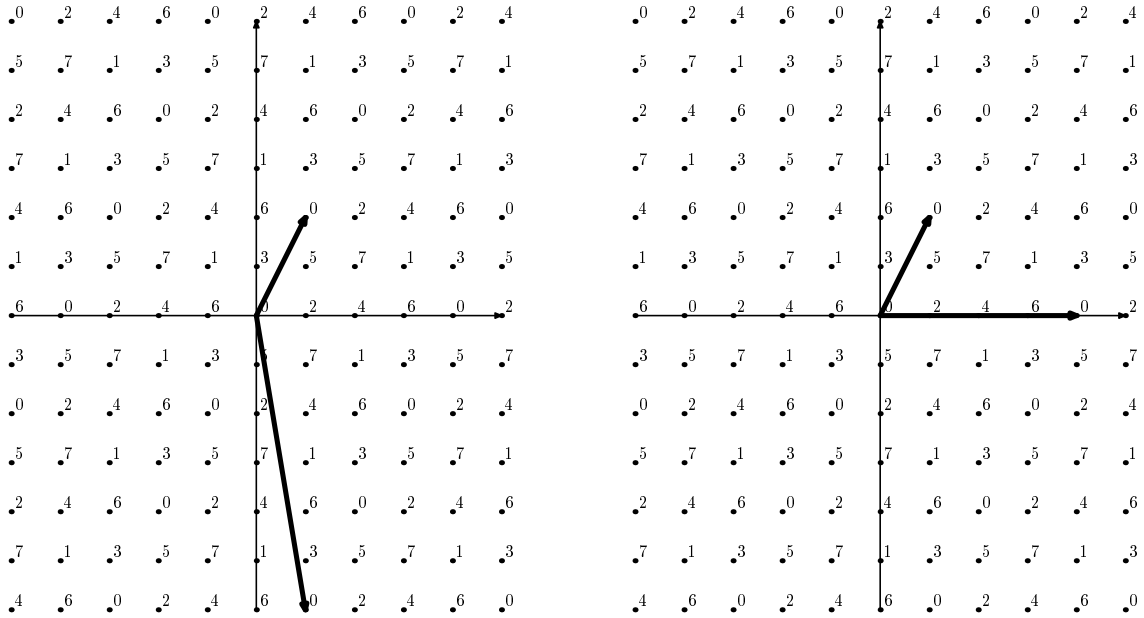
3.6.2 $\mathbb{Z}_2^2$ kot modul

Na mrežo $\mathbb{Z}_2^2$ lahko gledamo kot na dvodimenzionalni modul nad celimi števili, saj je množica $\mathbb{Z}_2^2$ Abelova grupa za običajno seštevanje parov celih števil in zaprta za množenje s celoštevilskimi skalarji. Baza modula $\mathbb{Z}_2^2$ je množica $\{(1,0),(0,1)\}$. V tem kontekstu lahko na množico X gledamo kot na dvodimenzionalni podmodul.

Trditev 3.6-II. *Množica X je dvodimenzionalni podmodul v $\mathbb{Z}_2^2$.*

Dokaz. Poleg zaprtosti množice X za notranjo in zunanjo operacijo (to sta seštevanje in množenje s celimi števili) moramo pokazati še, da obstajata vsaj dva linearno neodvisna elementa množice X. Zaprtost množice X sledi iz dejstva, da je l linearna funkcija in zato za poljubna $x, x' \in X$ in poljuben $i \in \mathbb{Z}$ velja $l(x + x') = l(x) + l(x') = 0$ in $l(i * x) = i * l(x) = 0$, torej $x + x' \in X$ in $i * x' \in X$. Ker sta elementa $(n, 0)$ in $(0, n)$ linearno neodvisna in oba v množici X, je X res dvodimenzionalni podmodul modula $\mathbb{Z}_2^2$. □

Po analogiji z vektorskim prostorom bomo tudi elemente mreže $\mathbb{Z}_2^2$ včasih imenovali *vektorji*.



Slika 3.6-II: Bazi (a) $\{(1,2),(1,-6)\}$ in (b) $\{(1,2),(4,0)\}$ za modul X grafa $G(8;\ \pm 2,\pm 3)$

Velikost elementov modula $\mathbb{Z}_2^2$
Za merjenje velikosti elementov mreže $\mathbb{Z}_2^2$ bomo uporabili tako imenovano Manhattan normo (včasih imenovano tudi poštarska ali ℓ_1 norma), ki je za $(x_1,x_2)\in\mathbb{Z}_2^2$ definirana z

$$||(x_1,x_2)||_1 = |x_1|+|x_2|.$$

Velikost elementa mreže se tako ujema z "velikostjo" pripadajočega ekvivalenčnega razreda (to je z dolžino sprehodov, ki temu razredu pripadajo). Ker bomo v tem delu uporabljali samo ℓ_1 normo, bomo namesto $||.||_1$ pisali kar $||.||$.

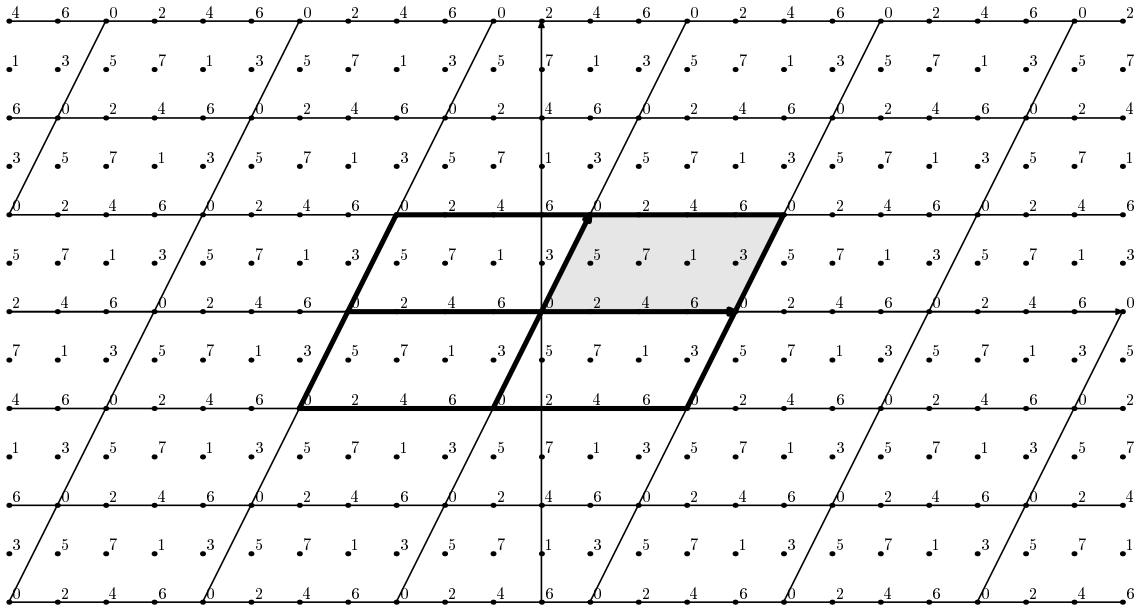
3.6.3 Baza modula X

Modulu X lahko poiščemo dvodimenzionalno bazo, to je množico dveh linearno neodvisnih vektorjev, ki ga napenjata. Na sliki 3.6-II vidimo dve različni bazi modula X za graf $G(8;\ \pm 2,\pm 3)$.

Naj bo $\{a,b\}$ baza modula X. **Osnovni bazni paralelogram** $[a,b)$ je množica točk v $\mathbb{Z}_2^2$, ki ležijo znotraj (in na polovici meje) paralelograma, ki ga napenjata vektorja a in b, oziroma,

$$[a,b) = \{x\in\mathbb{Z}_2^2;\ x=\alpha a+\beta b;\ \text{pri } 0\le\alpha,\beta<1\}.$$

Zaprti paralelogram (t.j. paralelogram, ki vsebuje vse svoje meje), bomo označevali z



Slika 3.6-III: Mreža $\mathbb{Z}_2^2$ tlakovana s paralelogrami baze $\{(1,2),(4,0)\}$ za modul X grafa $G(8;\ \pm 2,\pm 3)$; poudarjeno so prikazani štirje bazni paralelogrami, osnovni bazni paralelogram pa je zasenčen

$[a,b]$. Poleg osnovnega baznega paralelograma so za nas zanimivi še trije drugi paralelogrami in sicer tisti, ki jih dobimo, če osnovni bazni paralelogram vzporedno premaknemo v negativni smeri baznih vektorjev a, b in diagonale $a+b$. Z izrazom **bazni paralelogrami** bomo označevali paralelograme $[a,b)$, $[a,b)-a$, $[a,b)-b$ in $[a,b)-(a+b)$. Bazni paralelogrami tlakujejo območje okoli elementa $(0,0)$. Vsak drug paralelogram, torej vsak paralelogram oblike $[a,b)+sa+tb$ $(s,t\in\mathbb{Z})$, bomo preprosto imenovali **paralelogram baze** $\{a,b\}$. Celotna mreža $\mathbb{Z}_2^2$ je tlakovana s paralelogrami poljubne baze. Slika 3.6-III prikazuje mrežo $\mathbb{Z}_2^2$, tlakovano s paralelogrami baze $\{(1,2),(4,0)\}$ za modul X grafa $G(8;\ \pm 2,\pm 3)$; poudarjeno so prikazani štirje bazni paralelogrami, osnovni bazni paralelogram pa je zasenčen.

Ena od razlik med vektorskimi prostori in moduli se odraža v lastnostih baze. Pri vektorskih prostorih je vsaka linearno neodvisna množica, katere moč je enaka dimenziji prostora, baza, pri modulih pa to ne drži.

> Primer 3.6-II. Množica $\{(0,8),(8,0)\}\subset X$ v primeru grafa $G(11;\ \pm 2,\pm 3)$ ni baza, čeprav sta elementa med seboj neodvisna.

Poleg neodvisnosti morajo biti za elemente baze modula izpolnjeni še dodatni pogoji [29]. Med drugim mora veljati tudi, da je determinanta matrike baznih vektorjev po absolutni vrednosti enaka n ([6]). To pa z drugimi besedami pomeni, da je ploščina baznega paralelograma (pri poljubni bazi modula X) enaka n. Ker za vsak (polodprt) paralelogram s krajišči v $\mathbb{Z}_2^2$ in ploščino n velja, da vsebuje natanko n elementov množice $\mathbb{Z}_2^2$, n elemen-

tov vsebuje tudi bazni paralelogram $[a, b)$ poljubne baze $\{a, b\}$ modula X. Da imajo vsi ti elementi med seboj različne oznake (in je zato vsaka oznaka v baznem paralelogramu zastopana natanko enkrat), nam zagotavlja naslednja lema, katere natančen dokaz bralec najde v [54].

Lema 3.6-III. *Naj bo $\{a, b\}$ poljubna baza modula X. Potem paralelogram $[a, b)$ vsebuje natanko n točk in za vsak $w \in V_n$ obstaja tak $x \in [a, b)$, da $l(x) = w$.*

Bazni elementi
Vsak paralelogram baze $\{a, b\}$ torej vsebuje za vsako oznako $w \in V_n$ natanko en element, ki nosi to oznako. Za poljuben $w \in V_n$ bomo elementu, ki v osnovnem baznem paralelogramu nosi oznako w, rekli **osnovni bazni element** in ga označili z x_w. Velja

$$x_w \in [a, b) \cap X_w.$$

Z izrazom **bazni elementi** poimenujemo elemente z oznako w, ki so v štirih baznih paralelogramih. Množico baznih elementov bomo označili s C_1^w,

$$C_1^w = \{x_w, x_w - a, x_w - b, x_w - (a + b)\}.$$

Za vsak $w \in V_n$ so elementi z oznako w (torej elementi množic X_w) enakomerno porazdeljeni po mreži $\mathbb{Z}_2^2$. Če vzamemo poljuben element $x \in X_w$, lahko druge elemente množice X_w dobimo z vzporednimi premiki elementa x vzdolž vektorjev a in b (posamezen premik pomeni prehod iz paralelograma, kjer element trenutno je, v enega od sosednih paralelogramov). Če torej kot "začetni" element vzamemo osnovni bazni element, lahko vsak element $x \in X_w$ zapišemo v obliki

$$x = x_w + m_x a + n_x b \text{ pri } m_x, n_x \in \mathbb{Z}.$$

Ta zapis bomo v naslednjih poglavjih pogosto uporabljali.

3.6.4 Računanje baze modula X

Modul X ima običajno veliko različnih baz. Oglejmo si postopek za izračun ene izmed njih (algoritem 3.6-I).
Za prvi element baze izberimo najmanjšo ničlo na zgornjem delu ordinatne osi, $a = (0, n/\gcd(n, h_2))$. Ploščina baznega paralelograma mora biti enaka n, zato so za drugi element baze primerni le elementi oblike $b = (\gcd(n, h_2), x)$. Število x lahko izberemo poljubno, zagotoviti moramo le, da bo veljalo $b \in X$. Število x je torej poljubna rešitev kongruenčne enačbe

$$\gcd(n, h_2) * h_1 + x * h_2 \equiv 0 \pmod{n}.$$

Med vsemi temi rešitvami bomo mi izbrali tako, da bo "višina" baze čim manjša (t.j., da bo bazni paralelogram čim manj razpotegnjen). Element b bomo izbrali tako, da bo ležal

Algoritem 3.6-I	Baza(n, h_1, h_2)

Računanje baze modula X za graf $G(n;\ \pm h_1, \pm h_2)$

Baza(n, h_1, h_2) :

1: t_0:= poljubna rešitev enačbe $gcd(n, h_2)h_1 + th_2 \equiv 0 (\text{mod } n)$;

2: $t := t_0 \text{ mod } n/\text{gcd}(n, h_2)$;

3: $a := (0, n/gcd(n, h_2))$;

4: $b := (gcd(n, h_2), t - n/\text{gcd}(n, h_2))$;

5: **return** $\{a, b\}$.

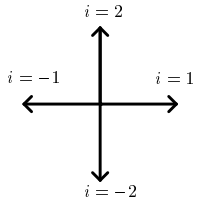
v četrtem kvadrantu in da bo njegova oddaljenost od abscisne osi čim manjša. Pri poljubni rešitvi t_0 zgornje enačbe število x določimo s

$$t = t_0 \text{ mod } \frac{n}{\gcd(n, h_2)} \text{ in } x = t - \frac{n}{\gcd(n, h_2)}.$$

Algoritem 3.6-I bazo modula X določi v času $\mathcal{O}(\log n)$.

Polravnine $\mathbb{Z}_i^2$

Pri reševanju nekaterih problemov, povezanih z dvokrožnimi grafi, se bomo morali omejiti le na eno od štirih polravnin mreže. Za $i = 1, -1, 2, -2$ bomo v ta namen z $\mathbb{Z}_i^2$ označili (po vrsti) desno, levo, zgornjo in spodnjo polravnino $\mathbb{Z}_2^2$.



Za točke (x_1, x_2) v $\mathbb{Z}_2^2$, ki ležijo v desni polravnini, velja $x_1 \geq 0$, za tiste v levi pa $x_1 \leq 0$. Podobno velja za točke zgornje in spodnje polravnine, kot je razvidno iz naslednje definicije.

Definicija 3.6-IV. *Naj bo $i \in I$. Z $\mathbb{Z}_i^2$ označimo množico točk $(x_1, x_2) \in \mathbb{Z}_2^2$, za katere velja sign(i) $* x_{|i|} \geq 0$.*

Definicija kot del posamezne polravnine $\mathbb{Z}_i^2$ šteje tudi elemente pripadajoče delilne osi, to je ordinate za $i = 1$ in -1 oziroma abscise za $i = 2$ in -2.

Pravokotni komplement elementov mreže

Podobno kot v vektorskem prostoru lahko tudi v modulih govorimo o "pravokotnosti" točk. Z izrazom **pravokotni komplement** elementa $x \in \mathbb{Z}_2^2$ bomo v nadaljevanju označevali tiste točke $y \in \mathbb{Z}_2^2$, ki imajo enako normo kot x in so nanj pravokotne. Za pravokotni komplement y elementa x torej velja $x \cdot y = 0$ in $||x|| = ||y||$ ($\cdot$ smo označili operacijo skalarnega produkta). Danemu neničelnemu elementu (x_1, x_2) vedno pripadata dva pravokotna komplementa in sicer

$$y' = (-x_2, x_1) \text{ in } y'' = (x_2, -x_1).$$

3.7 Predstavitev množice X_w

Kot smo videli v prejšnjem razdelku, elementi množic X_w niso "naključno razmetani" po $\mathbb{Z}_2^2$; njihovo razporeditev po mreži bi neformalno lahko opisali kot "simetrično" in "enakomerno". Natančneje se bomo z razporeditvijo elementov izbrane X_w ukvarjali v tem razdelku. Videli bomo, da lahko te množice predstavimo na dva načina in sicer kot unijo ciklov ali kot unijo trakov. Obe predstavitvi bomo uporabljali v poznejših poglavjih.

3.7.1 Ciklična predstavitev

Ustavimo se najprej pri pojmu ciklov. Naj bo $w \in V_n$ neka oznaka v mreži $\mathbb{Z}_2^2$ in naj bo $\{a, b\}$ izbrana baza modula X. Množico štirih baznih elementov $\{x_w, x_w - a, x_w - b, x_w - (a + b)\}$ bomo označili s C_1^w in imenovali **prvi cikel** množice X_w. Elementi množice C_1^w v $\mathbb{Z}_2^2$ so oglišča paralelograma. "Krožna" razporeditev elementov je vplivala na izbiro imena množice.

S premikanjem elementov množice C_1^w za $\pm a$, $\pm b$, $\pm(a + b)$ in $\pm(a - b)$ sestavimo novo množico, ki jo označimo s C_2^w in imenujemo **drugi cikel** množice X_w. Elementi množice C_1^w niso del množice C_2^w. Formalno C_2^w definiramo s

$$C_2^w = \{c \pm v;\ c \in C_1^w, v \in \{a, b, a + b, a - b\}\} - C_1^w.$$

Množica C_2^w ima, podobno kot množica C_1^w, obliko paralelograma. Vsebuje trikrat več točk kot osnovni bazni paralelogram in "enakomerno obkroža" množico C_1^w. Slika 3.7-I prikazuje primer mreže $\mathbb{Z}_2^2$, v kateri so posebej označeni elementi prvega in drugega cikla množice X_{10} grafa $G(11;\ \pm 3, \pm 4)$ pri izbrani bazi $\{a, b\} = \{(1, 2), (5, -1)\}$.

Podobno kot smo definirali drugi cikel, lahko (rekurzivno) definiramo tudi ostale. Za poljuben $i > 2$ bomo s C_i^w označevali i-ti cikel množice X_w, ki je definiran s

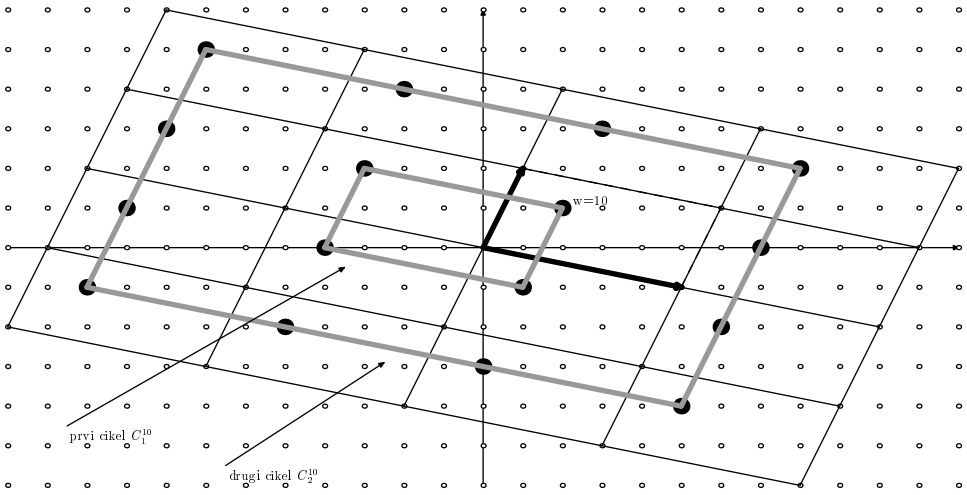
$$C_i^w = \{c \pm v;\ c \in C_{i-1}^w, v \in \{a, b, a + b, a - b\}\} - C_{i-1}^w - C_{i-2}^w.$$

Cikel C_i^w vsebuje $4 * (2i - 1)$ točk in "enakomerno obkroža" prejšnji cikel C_{i-1}^w. Vsak predstavnik množice X_w pripada enemu od ciklov $C_i^w, i = 1, 2, ...$ Rečemo, da množica X_w razpade na (disjunktne) cikle in pišemo

$$X_w = \bigcup_{i=1}^{\infty} C_i^w.$$

3.7.2 Vzporedna predstavitev

Druga predstavitev množice X_w je predstavitev z diskretnimi premicami. Videli smo namreč že, da lahko vsak element x množice X_w zapišemo v obliki $x = x_0 + ma + nb$,



Slika 3.7-I: Prvi in drugi cikel množice X_{10} v $G(11;\ \pm 3, \pm 4)$ pri $\{a, b\} = \{(1, 2), (5, -1)\}$. Elementi prvega in drugega cikla so označeni z debelimi pikami.

pri čemer je x_0 poljuben element množice X_w, m in n pa celi števili. Ob izbranem x_0 lahko torej X_w zapišemo kot unijo (diskretnih) premic skozi točke $x_0 + ma;\ (m \in \mathbb{Z})$ v smeri vektorja b ali kot unijo (diskretnih) premic skozi točke $x_0 + nb;\ (n \in \mathbb{Z})$ v smeri vektorja a. Izkazalo se bo, da je taka predstavitev množice X_w v določenih primerih bolj primerna kot predstavitev s cikli. Definirajmo sedaj še nekatere pojme za kasnejšo uporabo.

Definicija 3.7-I. *Naj bosta x in e točki v $\mathbb{Z}_2^2$. Diskretno premico mreže $\mathbb{Z}_2^2$ v smeri vektorja e, ki poteka skozi točko x, to je množico točk $x+ke, k \in \mathbb{Z}$, imenujemo* trak *in jo označimo s $t(x, e)$. Poltrak, ki ga dobimo kot presek traku $t(x, e)$ in polravnine $\mathbb{Z}_i^2$ označimo s $t^i(x, e)$.*

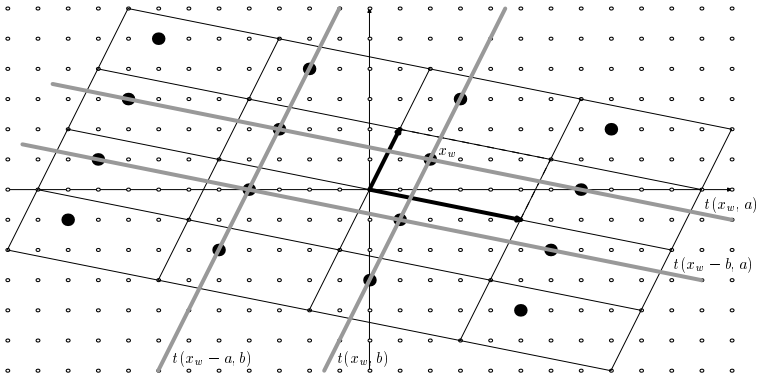
Posebej nas bodo zanimali trakovi, ki potekajo skozi štiri bazne elemente v smeri baznih vektorjev a in b (glej sliko 3.7-II). Množico, ki vsebuje točke teh trakov, bomo označili s T_w in ji rekli **množica baznih trakov**. Velja torej:

$$T_w = t(x_w, a) \cup t(x_w, b) \cup t(x_w - b, a) \cup t(x_w - a, b).$$

Ker je množica X_w zaprta za translaciji v smeri baznih vektorjev a in b ($a + X_w = b + X_w = X_w$), so vsi elementi množice T_w tudi elementi množice X_w, oziroma

$$T_w \subset X_w.$$

Zaradi preprostosti zapisa bomo pri izbrani bazi $\{a, b\}$ in osnovnem baznem elementu x_w trakove označevali tudi takole: traku $t(x_w + m_x a, b)$ bomo rekli b_m^w-trak ali preprosto **$\mathbf{b_m}$-trak**; podobno bomo traku $t(x_w + n_x b, a)$ rekli a_n^w oziroma **$\mathbf{a_n}$-trak**. Poljuben element $x = x_w + m_x a + n_x b \in X_w$ leži na sečišču trakov $t(x_w + n_x b, a)$ in $t(x_w + m_x a, b)$,



Slika 3.7-II: Bazni trakovi za graf $G(11;\ \pm 3, \pm 4)$ s pakirano bazo $\{a, b\} = \{(1, 2), (5, -1)\}$

oziroma, a_n in b_m trakov.

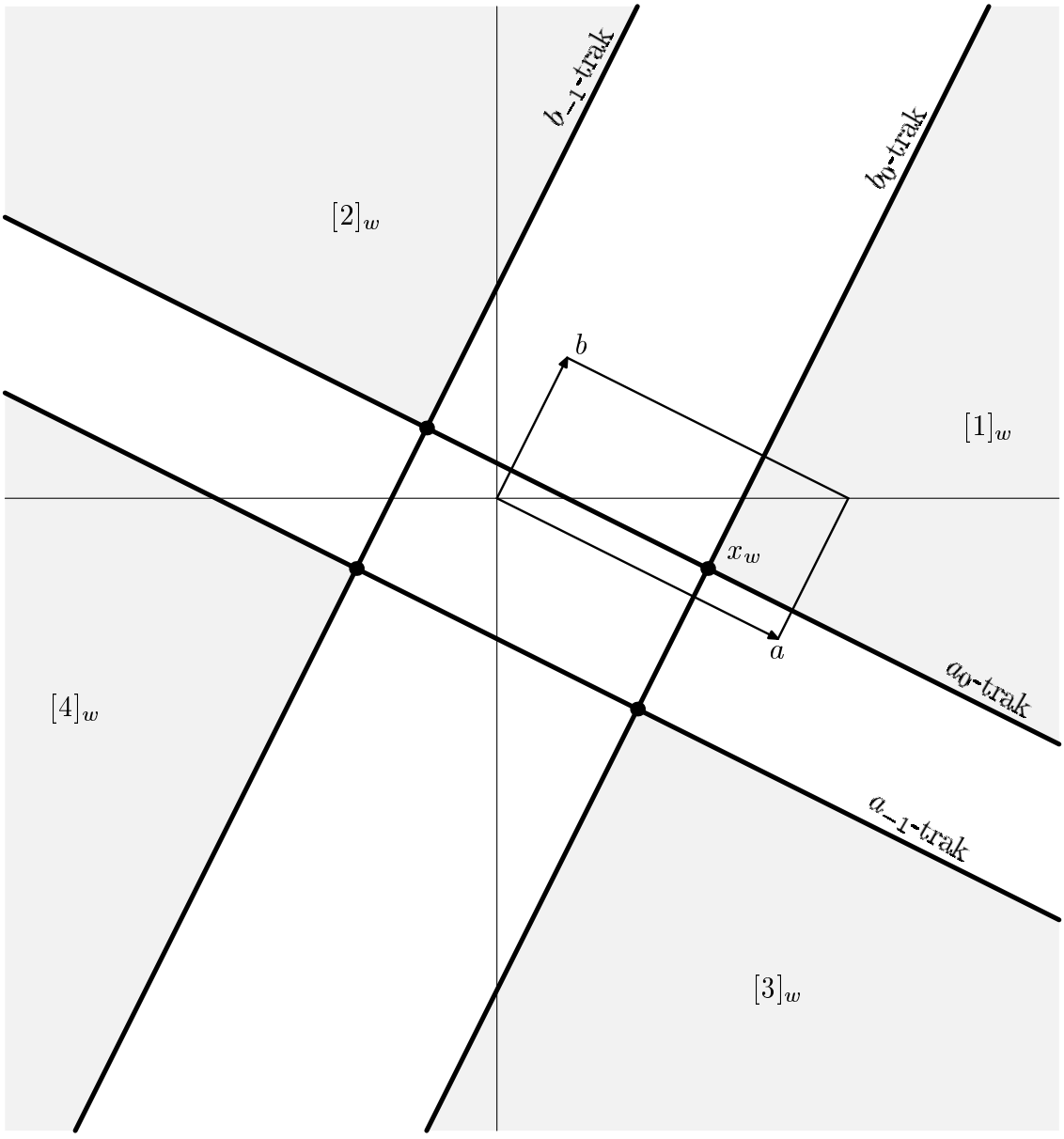
Če uporabimo preprostejši zapis, rečemo, da množico T_w sestavljajo trakovi $a_0,\ b_0,\ a_{-1}$ in b_{-1}. Ti trakovi razbijejo množico X_w v štiri disjunktne množice, oziroma na štiri **prekate** (glej sliko 3.7-III), ki jih bomo označili z $[\mathbf{1}]_\mathbf{w}$ (množica točk X_w, ki leži med trakovoma a_0 in b_0), $[\mathbf{2}]_\mathbf{w}$ (množica točk med trakovoma a_0 in b_{-1}), $[\mathbf{3}]_\mathbf{w}$ (množica točk med trakovoma a_{-1} in b_0) in $[\mathbf{4}]_\mathbf{w}$ (množica točk med trakovoma a_{-1} in b_{-1}). V delu $\mathbb{Z}_2^2$ med trakovoma a_0 in a_{-1} ter b_0 in b_{-1} ni predstavnikov množice X_w, saj bi bilo to v nasprotju z dejstvom, da v $[a, b)$ leži natanko en predstavnik množice X_w. Notranjost prekata $[P]_w$, to je množico točk prekata $[P]_w$ brez mejnih trakov, bomo označili s $(\mathbf{P})_\mathbf{w}$. Za $x \in X_w$, ki je oblike $x = x_w + m_x a + n_x b$, lahko na podlagi predznakov števil m_x in n_x določimo, v katerem prekatu leži. Tako na primer velja: $x \in [1]_w \iff (m_x \geq 0)$ in $(n_x \geq 0)$. Podobne karakteristike za druge prekate so zapisane v tabeli 3.7-I, kjer so za posamezen prekat navedeni tudi mejni trakovi in vrh prekata. V nadaljevanju besedila bomo črko w v oznaki prekata pogosto izpuščali (seveda kadar bo iz konteksta razvidno, kateri w imamo v mislih).

prekat	predznak m in n	mejna trakova	vrh prekata
$[1]_w$	$(m_x \geq 0)$ in $(n_x \geq 0)$	a_0, b_0	x_w
$[2]_w$	$(m_x < 0)$ in $(n_x \geq 0)$	a_0, b_{-1}	$x_w - a$
$[3]_w$	$(m_x \geq 0)$ in $(n_x < 0)$	a_{-1}, b_0	$x_w - b$
$[4]_w$	$(m_x < 0)$ in $(n_x < 0)$	a_{-1}, b_{-1}	$x_w - (a + b)$

Tabela 3.7-I: Karakteristike prekatov $[1]_w$, $[2]_w$, $[3]_w$, in $[4]_w$ množice X_w

Množico točk na baznih trakovih v polravnini $\mathbb{Z}_i^2$, to je množico $T_w \cap \mathbb{Z}_i^2$, bomo označili s T_w^i. Podobno bomo z X_w^i označili presek množic X_w in $\mathbb{Z}_i^2$. Seveda velja

$$T_w^i \subset X_w^i.$$



Slika 3.7-III: Štirje prekati množice X_w

Presek prekata $[P]_w$ s polravnino $\mathbb{Z}_i^2$ bomo označili s $[P]_w^i$, presek $(P)_w$ z $\mathbb{Z}_i^2$ pa s $(P)_w^i$.

3.8 Iskanje predstavnika množice X_w

Pri izbrani oznaki $w \in V_n$ lahko množico X_w opišemo tako, da izberemo poljubno bazo $\{a, b\}$ modula X ter poljuben element $x_0 \in X_w$ ter pišemo $X_w = \{x_0 + sa + tb;\ s, t \in \mathbb{Z}\}$. Če torej poznamo en element množice X_w, lahko tvorimo tudi druge; vprašanje je torej, kako poiskati prvi element te množice. Odgovor na to vprašanje se skriva v nadaljevanju tega razdelka, v katerem bomo pokazali tudi, kako ob izbrani bazi modula X poiščemo osnovni bazni element x_w. Odgovor na obe vprašanji bomo predstavili v splošni obliki, torej za poljuben $k \geq 2$.

3.8.1 Rešitev kongruenčne enačbe

Podobno kot je v primeru $k = 2$ vsak element množice X_w rešitev kongruenčne enačbe 3.3 pri $u = 0$ in $v = w$, je za splošen primer (t.j. pri poljubnem $k \geq 2$) vsak element množice X_w rešitev kongruenčne enačbe

$$x_1h_1 + x_2h_2 + \ldots + x_kh_k \equiv w \pmod{n}.$$

Rešitev te enačbe poiščemo s k-kratno uporabo (razširjenega) Evklidovega algoritma [35, 42], ki za dani par (x, y) poleg največjega skupnega delitelja $g = \gcd(x, y)$ poišče tudi tak par (p, q), da velja $px + qy = g$. Znano je, da Evklidov algoritem za izračun rezultata potrebuje $\mathcal{O}(\log n)$ časa. Zaradi preprostosti zapisa bomo v nadaljevanju z g_i označevali največji skupni delitelj števil $h_1, \ldots, h_i$, torej,

$$g_i := \gcd(h_1, \ldots, h_i) \text{ za } i = 1, 2, \ldots k.$$

Iskanje partikularne rešitve zgornje kongruenčne enačbe je ekvivalentno iskanju rešitve diofantske enačbe

$$x_1h_1 + x_2h_2 + \ldots + x_kh_k + Nn = w. \tag{3.5}$$

Enačbe (3.5) se bomo lotili tako, da bomo v k zaporednih korakih z združevanjem spremenljivk zmanjševali njeno kompleksnost, dokler nam ne bo v zadnjem koraku ostala le še preprosta enačba dveh spremenljivk. Združevanje spremenljivk bomo opravili v smeri od leve proti desni. V prvem koraku bomo združili spremenljivki h_1 in h_2 in sicer tako, da bomo z Evklidovim algoritmom rešili enačbo

$$1 \quad : \quad y_1h_1 + z_1h_2 = g_2.$$

Združeni spremenljivki h_1 in h_2 lahko v nadaljevanju nastopata kot ena sama in enačbo 3.5 lahko prepišemo v obliki

$$y_2g_2 + x_3h_3 + \ldots + x_kh_k + Nn = w.$$

S tem smo zmanjšali število spremenljivk prvotne enačbe na tak način, da lahko iz rešitve $(y_2, x_3, \ldots, x_k, N)$ nove enačbe neposredno preberemo rešitev prvotne enačbe, ki se glasi $(y_2y_1, y_2z_1, x_3, \ldots, x_k, N)$. S postopkom združevanja spremenljivk nadaljujemo tako, da po vrsti rešimo naslednje enačbe:

$$\begin{array}{ll}
2 & : \quad y_2g_2 + z_2h_3 = g_3, \\
3 & : \quad y_3g_3 + z_3h_4 = g_4, \\
\ddots & \qquad\quad \ddots \\
k-1 & : \quad y_{k-1}g_{k-1} + z_{k-1}h_k = g_k, \\
k & : \quad y_kg_k + z_kn = 1.
\end{array}$$

Dodatni komentarji k zgornjim enačbam niso potrebni, saj smo jih dobili na enak način kot prvo. Omenimo le, da je enica na desni strani zadnje enačbe posledica predpostavke $\gcd(n, h_1, h_2, \ldots, h_k) = 1$. Rešitev prvotne enačbe zdaj dobimo tako, da rešitev zadnje enačbe vstavimo v predzadnjo, to nato vstavimo v predpredzadnjo in tako naprej, vse dokler ne pridemo do začetka:

$$\begin{array}{c}
y_kg_k + z_kn = 1 \\
\Downarrow \\
y_k(y_{k-1}g_{k-1} + z_{k-1}h_k) + z_kn = 1 \\
\| \\
y_ky_{k-1}g_{k-1} + y_kz_{k-1}h_k + z_kn = 1 \\
\Downarrow \\
y_ky_{k-1}(y_{k-2}g_{k-2} + z_{k-2}h_{k-1}) + y_kz_{k-1}h_k + z_kn = 1 \\
\| \\
y_ky_{k-1}y_{k-2}g_{k-2} + y_ky_{k-1}z_{k-2}h_{k-1} + y_kz_{k-1}h_k + z_kn = 1 \\
\vdots \\
y_ky_{k-1}\ldots y_1h_1 + z_1y_ky_{k-1}\ldots y_2h_2 + \cdots + z_{k-2}y_ky_{k-1}h_{k-1} + z_{k-1}y_kh_k + z_kn = 1
\end{array}$$

Če z f_i označimo produkt števil y_j

$$f_i := \Pi_{j=i}^{k} y_j$$

in z $z_0 = 1$, lahko rešitve diofantske enačbe (3.5) zapišemo v obliki

$$x_i = wz_{i-1}f_i.$$

V algoritmu 3.8-I, ki povzema pravkar opisani postopek za reševanje diofantske enačbe (3.5), smo z ExtendedGCD(x, y) označili funkcijo, ki vrne rezultat razširjenega Evklidovega algoritma v obliki $\{g, \{p, q\}\}$, kjer je g največji skupni delitelj števil x in y, par $\{p, q\}$ pa tak, da velja $px + qy = g$.

Algoritem 3.8-I DIOF($n, h_1, \ldots, h_k, w$)

Reševanje diofantske enačbe $x_1h_1 + x_2h_2 + \ldots + x_kh_k + Nn = w$

DIOF($n, h_1, \ldots, h_k, w$) :

1: $g_1 := h_1;\ h_{k+1} := n;$
2: **for** $i := 1$ **to** k **do**
3: $\{g_{i+1}, \{y_i, z_i\}\} :=$ ExtendedGCD(g_i, h_{i+1});
4: $z_0 := 1;\ f := 1;$
5: **for** $i := k$ **to** 1 **do begin**
6: $f := f * y_i$**;**
7: $x_i := w * z_{i-1} * f$**;**
8: **end;**
9: **return** $(x_1, x_2, \ldots, x_k, z_k)$.

3.8.2 Iskanje osnovnega baznega elementa

Osnovni bazni element x_w je element množice X_w, ki leži v baznem paralelepipedu. Z drugimi besedami: x_w je tisti element množice X_w, katerega VSI koeficienti pri zapisu z baznimi vektorji so večji ali enaki 0 in strogo manjši od 1. Osnovni bazni element poiščemo tako, da vzamemo poljubni element množice X_w (mi bomo vzeli element x, ki ga dobimo kot partikularno rešitev diofantske enačbe 3.5 – glej zgoraj), ga izrazimo v bazi in koeficientom tega izraza odštejemo njihov celi del (zaokroževanje s funkcijo $\lfloor . \rfloor$); tako dobljeni koeficienti predstavljajo koeficiente pri zapisu elementa x_w v dani bazi.

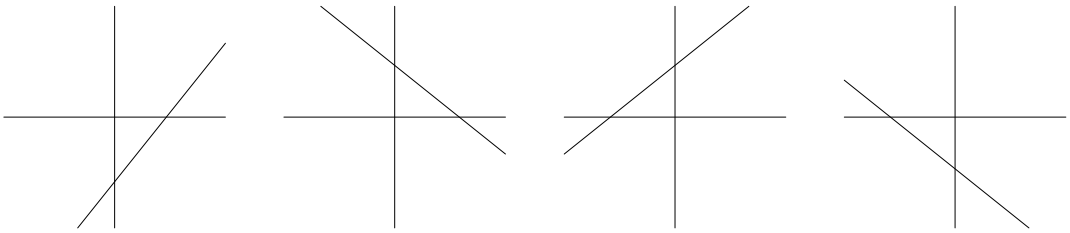
Naj bo B bazna matrika modula X, to je matrika, katere stolpci so vektorji $b_1, b_2, \ldots, b_k$. Označimo z α vektor koeficientov pri zapisu elementa x v bazi; natančneje,

$$\text{če } x = B\alpha = \alpha_1 b_1 + \alpha_1 b_1 + \ldots + \alpha_k b_k, \text{ potem } \alpha = B^{-1}x.$$

Če nadalje uvedemo oznako $\lfloor \alpha \rfloor = (\lfloor \alpha_1 \rfloor, \lfloor \alpha_2 \rfloor, \ldots, \lfloor \alpha_k \rfloor)$, potem velja

$$x_w = x - B(\lfloor \alpha \rfloor) = x - B(\lfloor B^{-1}x \rfloor).$$

Časovno najbolj potratna operacija opisanega postopka je izračun inverza matrike B, ki ga lahko opravimo v času $\mathcal{O}(k^3)$. Skupna časovna zahtevnost računanja elementa x_w je torej $\mathcal{O}(k^3)$.



Slika 3.9-I: Štirje načini vstopa premice $x + s\hat{e}$ v desno polravnino

3.9 Projekcije v celoštevilski mreži

V tem razdelku bomo predstavili formule za izračun najmanjšega elementa danega traku ter danega poltraku v celoštevilski mreži, ki jih potrebujemo pri računanju posebnega tipa najkrajših poti v krožnih grafih (t.i. omejenih najkrajših poti, glej razdelek 3.12)

3.9.1 Minimalna projekcija v $\mathbb{Z}_2^2$

Najmanjši element danega poltraku $t^i \subset \mathbb{Z}_2^2$ je taka točka $x \in t^i$, da za vsak $x' \in t^i$ velja $||x'|| \geq ||x||$. Najmanjši element bomo imenovali minimalna projekcija, saj ga dobimo tako, da vzamemo poljubno točko na traku in izvedemo projekcijo v smeri traku do minimalne točke v pripadajoči polravnini (t.j. prištevamo ali odštevamo smerni vektor traku, dokler ne pridemo do točke z minimalno normo).

Definicija 3.9-I. *Naj bosta* $x, e \in \mathbb{Z}_2^2$ *in* $i \in \{1, -1, 2, -2\}$. *Najmanjši element poltraku* $t^i(x, e)$ *imenujemo* minimalna projekcija točke x v smeri e v polravnino $\mathbb{Z}_i^2$ *in označimo s* $P^i(x, e)$.

Ker se projekcija v zgornjo (oziroma spodnjo) polravnino razlikuje od projekcije v levo (oziroma desno) polravnino, potrebujemo 4 različne formule. V nadaljevanju bomo zato najprej izpeljali formulo za izračun minimalne projekcije točke x v smeri e v desno polravnino ($i = 1$), nato bomo formulo zapisali v splošni obliki, ki bo primerna za vsak i.

Naj bo torej $i = 1$. Če je $e_1 = 0$, potem projekcija vzdolž vektorja e v desno polravnino ni mogoča, zato predpostavimo, da je $e_1 \neq 0$. Namesto projekcije v smeri e bomo obravnavali projekcijo v smeri $\hat{e} = (\hat{e}_1, \hat{e}_2) =$ sign$(e_1) * e$ – vektor, vzdolž katerega bomo projecirali, je vektor prvega oziroma četrtega kvadranta (na ta način poenotimo izpeljavo za oba primera, to je, ko $e_1 \leq 0$ in ko $e_2 \geq 0$).
Minimalna projekcija $P^1(x, e)$ je natančno določena s celim številom $k \in \mathbb{Z}$, za katerega velja

1. $x + k\hat{e} \in \mathbb{Z}_1^2$ in

2. za vsak $t \in \mathbb{Z}$, za katerega je $x + t\hat{e} \in \mathbb{Z}_1^2$, velja: $||x + k\hat{e}|| \leq ||x + t\hat{e}||$.

Trak $x + s\hat{e}, s \in \mathbb{R}$, ki seka ordinatno os pri $s = -x_1/\hat{e}_1$, lahko vstopi iz leve v desno polravnino na dva načina in sicer:

a) po vstopu se abscisni osi približuje (sliki 3.9-I(a) in 3.9-I(b)) ali

b) po vstopu se abscisni osi ne približuje (torej se od nje oddaljuje ali pa poteka vzporedno z njo) (sliki 3.9-I(c) in 3.9-I(d)).

Primer a) se zgodi bodisi, ko naraščajoča premica ($\hat{e}_2 > 0$) seka ordinatno os v spodnji polravnini ($x_2 + (-x_1/\hat{e}_1)\hat{e}_2 < 0$) (slika 3.9-I(a)) bodisi, ko padajoča premica ($\hat{e}_2 < 0$) seka ordinatno os v zgornji polravnini ($x_2 + (-x_1/\hat{e}_1)\hat{e}_2 > 0$) (sliki 3.9-I(b)). Oziroma, če preuredimo pravkar napisane neenačbe, ko $x_2/\hat{e}_2 < x_1/\hat{e}_1$. V nasprotnem primeru imamo opravka s primerom b). Glede na naklon premice lahko v primeru a) ločimo dva načina približevanja abscisni osi

a.i) približevanje, ki hkrati manjša tudi normo (to se zgodi le, če $\hat{e}_1 < |\hat{e}_2|$) in

a.ii) približevanje, ki norme ne manjša (torej jo veča ali ohranja).

Zdaj, ko smo problem podrobneje razčlenili, lahko zapišemo še formule za izračun minimalne projekcije. V primerih a.ii) in b) se z oddaljevanjem od ordinatne osi norma elementov na premici ne manjša, zato bo najmanjši element tega traku v desni polravnini tisti, ki je najbližji ordinatni osi, oziroma $P^1(x, e)$=$x + \lceil -x_1/\hat{e}_1 \rceil \hat{e}$. V primeru a.i) pa z oddaljevanjem od ordinate elementom na premici norma pada - vendar samo toliko časa, dokler premica ne seka abscise (to se zgodi pri $s = -x_2/\hat{e}_2$), potem začne spet naraščati. Element z najmanjšo normo bo tisti element premice, ki je najbližji abscisni osi, oziroma $P^1(x, e)$=$x + \lfloor -x_2/\hat{e}_2 \rceil \hat{e}$. Napisane ugotovitve lahko združimo v naslednjem algoritmu za izračun minimalne projekcije $P^1(x, e)$:

- če $e_1 = 0$, potem projekcija ni mogoča $\rightarrow$ konec postopka,
- naj bo $\hat{e} =$ sign$(e_1) * e$,
- **if** $(\hat{e}_1 < |\hat{e}_2|)$ **and** $(\frac{x_2}{\hat{e}_2} < \frac{x_1}{\hat{e}_1})$
  then $k := \lfloor -\frac{x_2}{\hat{e}_2} \rceil$ **else** $k := \lceil -\frac{x_1}{\hat{e}_1} \rceil$,
- $P^1(x, e)$:=$x + k\hat{e}$.

Izpeljava formul za minimalno projekcijo v $\mathbb{Z}_2^2$ poteka podobno kot pravkar prikazan postopek za minimalno projekcijo v $\mathbb{Z}_1^2$, le da se v vseh formulah zamenja pomen indeksov 1 in 2. Previdnejši pa je treba biti pri projekcijah v $\mathbb{Z}_{-1}^2$ in $\mathbb{Z}_{-2}^2$. Tu se namreč spremeni kriterij za "pravilnost vstopa" v levo (oziroma spodnjo) polravnino (namesto pogoja $x_2/\hat{e}_2 < x_1/\hat{e}_1$, ki smo ga potrebovali pri $i = 1$, pri $i = -1$ potrebujemo obrnjen pogoj $x_2/\hat{e}_2 > x_1/\hat{e}_1$), poleg tega moramo pri zaokroževanju števila k namesto funkcije $\lceil \, . \, \rceil$ uporabiti funkcijo $\lfloor \, . \, \rfloor$ (ker je $\hat{e}$ usmerjen v desno polravnino, je $x + \lceil -x1/\hat{e}_1 \rceil \hat{e}$ prvi element diskretnega traku v desni polravnini, $x + \lfloor -x1/\hat{e}_1 \rfloor \hat{e}$ pa zadnji v levi; podobno velja za zgornjo in spodnjo polravnino). Pri zapisu algoritma 3.9-I smo upoštevali omenjene razlike in podali splošno formulo za izračun minimalne projekcije $P^i(x, e)$.

Algoritem 3.9-I $P^i(x,e)$

Računanje minimalne projekcije $P^i(x,e)$ pri $k=2$

$P^i(x,e)$:

1: **if** ($e_{|i|}=0$) **then return** null; // projekcija v $\mathbb{Z}_i^2$ ni mogoča

2: $\hat{e} :=$ sign($e_{|i|}$) $* e$; $j := 3-|i|$;

3: **if** ($i<0$) **then** // katero zaokroževalno funkcijo bomo uporabili?
$f(x) := \lfloor x \rfloor$
else
$f(x) := \lceil x \rceil$;

4: **if** (($\hat{e}_{|i|} < |\hat{e}_j|$) **and** (sign($i$) $* \frac{x_j}{\hat{e}_j} <$ sign(i) $* \frac{x_{|i|}}{\hat{e}_{|i|}}$)) **then**
$k := \lfloor -\frac{x_j}{\hat{e}_j} \rceil$
else
$k := f(-\frac{x_{|i|}}{\hat{e}_{|i|}})$;

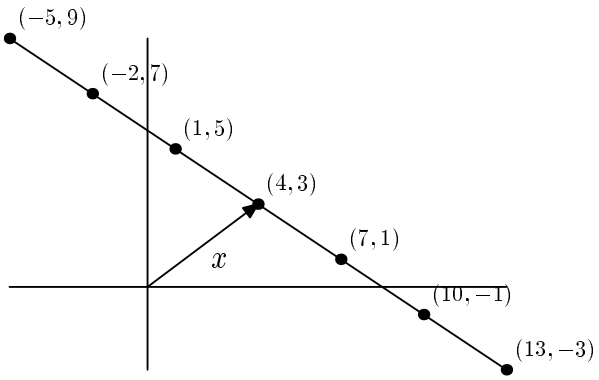
5: **return** $x + k\hat{e}$.

3.9.2 Najmanjši element na traku v $\mathbb{Z}_2^2$

V prejšnjem razdelku smo definirali pojem minimalne projekcije. Vprašajmo se, kako bi pridobljeno znanje uporabili za izračun najmanjšega elementa na danem traku. Definirajmo najprej problem malo natančneje.

Definicija 3.9-II. *Naj bosta x in e elementa celoštevilske mreže $\mathbb{Z}_2^2$ in $e \neq (0,0)$. Najmanjši element traku $t(x,e)$, ki ga označimo s $P(x,e)$, je element oblike $x+ke$ z najmanjšo ℓ_1 normo.*

Najmanjši element traku lahko izračunamo z dvakratno uporabo minimalne projekcije. Če namreč za $i \in \{1,2\}$ velja $e_i \neq 0$, potem je $P(x,e)$ enak manjšemu izmed elementov



Slika 3.9-II: Minimalni element na traku $t((4,3),(3,-2))$ je element (1,5)

$P^i(x,e)$ in $P^{-i}(x,e)$.

Primer 3.9-I. Naj bo $x=(4,3)$ in $e=(3,-2)$ (glej sliko 3.9-II). Ker sta obe koordinati vektorja e neničelni, lahko najmanjši element na traku $t(x,e)$ izračunamo na dva načina in sicer z minimalnima projekcijama v polravnini $\mathbb{Z}^2_1$ in $\mathbb{Z}^2_{-1}$ ter z minimalnima projekcijama v polravnini $\mathbb{Z}^2_2$ in $\mathbb{Z}^2_{-2}$. Oglejmo si oba izračuna. Minimalna projekcija v $\mathbb{Z}^2_{-1}$ je prvi element po vstopu traku v levo polravnino, saj se norma elementov traku z oddaljevanjem od ordinatne osi veča, torej $P^{-1}(x,e)$=(-2,7). Podobno je projekcija $\mathbb{Z}^2_1$ enaka prvemu elementu v desni polravnini, torej $P^1(x,e)$=(1,5). Najmanjši element traku $t(x,e)$ je po prvem izračunu enak elementu (1,5). Pri uporabi projekcije v $\mathbb{Z}^2_2$ moramo biti malo bolj previdni. Po vstopu v zgornjo polravnino se elementom traku norma manjša, dokler trak ne seka ordinatne osi. Minimalna projekcija $P^2(x,e)$ bo zato enaka manjšemu izmed elementov traku, ki sta najbližja ordinatni osi, (-2,7) in (1,5). Za projekcijo v $\mathbb{Z}^2_{-2}$ pa ponovno velja, da je enaka prvemu elementu na traku po vstopu v spodnjo polravnino, torej $P^{-2}(x,e)$=(10,-1). Iz vsega povedanega sledi, da tudi drugi izračun vrne isti rezultat, to je $P(x,e)$=(1,5).

Pri računanju minimalne projekcije smo ločili več primerov - glede na način vstopa traku v pripadajočo polravnino in glede na naklon smernega vektorja. V vseh primerih se je izkazalo, da rešitev leži pri sečišču traku in ene od koordinatnih osi. Ker je računanje najmanjšega elementa traku vezano na računanje minimalne projekcije, velja podoben sklep: najmanjši element na $t(x,e)$ je eden od (največ) štirih elementov tega traku, ki ležijo "pred" in "po" sečišču tega traku in koordinatnih osi. Če je katera od koordinat vektorja e enaka nič, potem sta kandidata za najmanjši element traku samo še dva; število kandidatov se zmanjša tudi takrat, ko točka traku leži na eni od koordinatnih osi. Število kandidatov za minimalni element traku se tako giblje nekje med 1 in 4. Natančen postopek za izračun elementa $P(x,e)$ je podan v algoritmu 3.9-II, v katerem najprej izračunamo elementa blizu sečišča traku in ordinatne osi (vrstica 2), nato elementa blizu sečišča traku in abscisne osi (vrstica 3), na koncu pa med izračunanimi elementi izberemo tistega, ki ima najmanjšo normo. Časovna zahtevnost tega algoritma je $\mathcal{O}(1)$.

Algoritem 3.9-II $P(x,e)$

Računanje najmanjšega elementa traku $t(x,e)$ pri $k=2$

$P(x,e)$:

1: S:={};

2: **if** ($e_1 \neq 0$) **then**
$S := S \cup \{x + \lfloor -\frac{x_1}{e_1} \rfloor e,\ x + \lceil -\frac{x_1}{e_1} \rceil e\}$;

3: **if** ($e_2 \neq 0$) **then**
$S := S \cup \{x + \lfloor -\frac{x_2}{e_2} \rfloor e,\ x + \lceil -\frac{x_2}{e_2} \rceil e\}$;

4: **return** $\min_{\ell_1} S$

3.9.3 Minimalna projekcija v k-dimenzijah

Podobno kot smo definirali in izpeljali formule za minimalno projekcijo v celoštevilski ravnini $\mathbb{Z}_2^2$, bomo to zdaj storili še za splošen primer, to je v k-dimenzionalni mreži $\mathbb{Z}^k$ pri poljubnem $k \geq 2$. Definicijo minimalne projekcije napišimo znova, tokrat prilagojeno k-dimenzionalnemu izrazoslovju.

Definicija 3.9-III. *Naj bosta $x, e \in \mathbb{Z}^k$ in $i \in I$. Najmanjši element poltraku $t^i(x,e)$ imenujemo* minimalna projekcija točke x v smeri e v polprostor $\mathbb{Z}_i^k$ *in označimo z* $P^i(x,e)$.

V tej definiciji smo z I, kot smo to navajeni v k-dimenzionalnem primeru, označili množico $I = \{-k, -k+1, \ldots, -1, 1, \ldots, k-1, k\}$, z $\mathbb{Z}_i^k$ pa k-dimenzionalni polprostor, ki ga dobimo kot presek k-dimenzionalne mreže $\mathbb{Z}^k$ in i-tega polprostora,

$$\mathbb{Z}_i^k := \{x \in \mathbb{Z}^k;\ x_{|i|} * \text{ sign}(i) \geq 0\}.$$

S $t^i(x,e)$ smo označili presek traku $t(x,e)$=$\{x + te;\ t \in \mathbb{Z}\}$ in polprostora $\mathbb{Z}_i^k$. Po analogiji s tri-dimenzionalnim primerom bomo tudi v splošnem primeru $(k-1)$-dimenzionalne koordinatne prostore imenovali *koordinatna ravnina*.

V dvodimenzionalnem primeru smo poiskali eksplicitno formulo za izračun minimalne projekcije, v k-dimenzionalnem primeru pa je to, zaradi večjega števila posebnih primerov, mnogo težja naloga. Zato bomo namesto eksplicitne formule predstavili iterativni postopek, ki minimalno projekcijo izračuna v k zaporednih korakih. Pri računanju minimalne projekcije elementa x vzdolž e v $\mathbb{Z}_i^k$ bomo z $\hat{e}$ označili vektor, ki je vzporeden vektorju e in kaže v i-ti polprostor,

$$\hat{e} := \text{ sign}(i * e_{|i|}) * e.$$

Če $e_i = 0$, projekcija vzdolž e ni mogoča, v vseh drugih primerih pa moramo med elementi traku $t^i(x,\hat{e})$ poiskati element z najmanjšo normo. Trak $t(x,\hat{e})$ "pride" v i-ti polprostor, ko velja $x_i + k\hat{e}_i = 0$, zato

$$t^i(x,e) = \left\{ x + k\hat{e};\ \text{pri } k \geq \left\lceil -\frac{x_{|i|}}{\hat{e}_{|i|}} \right\rceil \right\}.$$

Ker vemo, da trak doseže svoj minimum v bližini sečišča z eno od koordinatnih ravnin (to je pri $x_j + k\hat{e}_j = 0$ za $j \in I^+$), minimalno projekcijo poiščemo med elementi

$$x + \left\lfloor -\frac{x_{|j|}}{\hat{e}_{|j|}} \right\rfloor * \hat{e},$$

pri $j \in I^+$, za katere velja $\frac{-x_j}{\hat{e}_j} \geq \left\lceil -\frac{x_{|i|}}{\hat{e}_{|i|}} \right\rceil$. Natančen postopek za izračun minimalne projekcije je prikazan v algoritmu 3.9-III. Časovna zahtevnost algoritma je $\mathcal{O}(k)$.

Algoritem 3.9-III $P^i(x,e)$

Računanje minimalne projekcije $P^i(x,e)$ v k-dimenzijah

$P^i(x,e)$:

1: **if** $(e_{|i|}=0)$ **then return** null; // projekcija v $\mathbb{Z}_i^k$ ni mogoča
2: $\hat{e} :=$ sign$(i * e_{|i|}) * e$;
3: $k_0 := \left\lceil -\frac{x_{|i|}}{\hat{e}_{|i|}} \right\rceil$; $minP := x + k_0\hat{e}$;
4: **for** $j=1$ **to** k, $j \neq i$ **do**
5: **if** $(e_j \neq 0)$ **and** $(-\frac{x_j}{\hat{e}_j} > k)$ **and** $(||x + \left\lfloor -\frac{x_j}{\hat{e}_j} \right\rfloor * \hat{e}|| < ||minP||)$
6: **then** $minP := x + \left\lfloor -\frac{x_j}{\hat{e}_j} \right\rfloor * \hat{e}$;
7: **return** $minP$.

3.10 Pakirana baza

Videli smo, da za modul X, ki pripada nekemu grafu $G(n;\ \pm h_1, \pm h_2)$, obstaja več različnih baz. Izkazalo se bo, da vse baze niso enako uporabne za reševanje problemov. Pri nekaterih problemih bomo potrebovali bazo, katere diagonali (vsota in razlika baznih vektorjev) nista krajši od vektorjev baze. Za tako bazo bomo rekli, da je pakirana.

Definicija 3.10-I. *Bazo $\{a,b\}$ modula X, za katero velja*

$$\max\{||a||, ||b||\} \leq \min\{||a+b||, ||a-b||\},$$

imenujemo pakirana baza.

Iterativni algoritem za izračun pakirane baze
Pojem pakirane baze sta uvedla Žerovnik in Pisanski v [54], kjer sta dokazala, da pakirana baza vedno obstaja, in podala iterativni algoritem 3.10-I za njen izračun.
Osnovna ideja iterativnega dela tega algoritma (vrstice 6-10) je sledeča: če baza ni pakirana, zamenjaj "predolg" bazni vektor s "prekratko" diagonalo; postopek ponavljaj, dokler ne prideš do pakirane baze. Časovna zahtevnost algoritma je $\mathcal{O}(\log n)$ [54].

3.10.1 Eksplicitna formula za izračun pakirane baze

Ker smo pri našem delu pakirano bazo pogosto uporabljali, smo želeli poiskati algoritem, s katerim bi jo izračunali na ne-iterativni način (brez zanke). Žal smo se temu cilju približali le delno. Iskani algoritem smo sicer našli (imenujemo ga algoritem FPB, glej algoritem 3.10-II), toda izkazalo se je, da rezultat, ki ga vrača, ni povsem pravilen. Algoritem FPB smo preverili na velikem številu konkretnih krožnih grafov. Prvi rezultati

Algoritem 3.10-I PakiranaBaza(n, h_1, h_2)

Izračun pakirane baze modula X za graf $G(n;\ \pm h_1, \pm h_2)$

PakiranaBaza(n, h_1, h_2) :

1: $t_0 :=$ poljubna rešitev enačbe $gcd(n, h_2)h_1 + th_2 \equiv 0 (\text{mod } n)$;
2: $t := t_0 \text{ mod } n/\text{gcd}(n, h_2)$;
3: $a_0 := (x_0, y_0) := (0, n/gcd(n, h_2))$;
4: $a_1 := (x_1, -y_1) := (gcd(n, h_2), t - n/\text{gcd}(n, h_2))$;
5: $i := 1$;
6: **while** $x_i < y_i$ **do begin**
7: $\quad x_{i+1} := x_{i-1} + x_i \lfloor y_{i-1}/y_i \rfloor$;
8: $\quad y_{i+1} := y_{i-1} \text{ mod } y_i$;
9: $\quad i := i + 1$
10: **end**;
11: **if** $\min\{||a_i + a_{i-1}||, ||a_i - a_{i-1}||\} < \max\{||a_i||, ||a_{i-1}||\}$
12: $\quad$ **then** $B := \{\min\{a_i, a_{i-1}\},\ \min\{a_i + a_{i-1}, a_i - a_{i-1}\}\}$
13: $\quad$ **else** $B := \{a_i, a_{i-1}\}$;
14: **return** B.

Opomba. Z a_i smo v algoritmu označevali vektor $(x_i, (-1)^i y_i)$.

(začeli smo pri grafih z majhnim številom točk) so bili obetavni. Žal pa se je pri večjem številu točk zalomilo – tam je algoritem odpovedal, saj baza, ki smo jo dobili z njegovo uporabo, ni bila pakirana. Ker smo se želeli prepričati o domnevi, da FPB vrača pravilen rezultat vsaj za majhne n, smo sistematično pregledali vse krožne grafe s petimi, šestimi, sedmimi ... točkami. Ugotovili smo naslednje: algoritem FPB vrne pravi rezultat za VSE krožne grafe, ki imajo manj kot 115 točk (torej za "prvih" 58.000 krožnih grafov). Algoritem je odpovedal na grafu $G(115;\ \pm 2, \pm 27)$. Da bi ugotovili, s kolikšno verjetnostjo FPB vrača pravi rezultat, smo ga preverili še na 50.000 naključno izbranih krožnih grafih s številom točk med 115 in 10.000. Ugotovili smo, da je na tej testni množici algoritem v 75% vrnil pravi rezultat, v drugih 25% pa je odpovedal. Ugotovili smo tudi, da se z večanjem števila n zanesljivost algoritma manjša. Tako je, na primer, med 1.000 grafi z $n = 200$ nezanesljivost manj kot 1%, med 1.000 grafi z $n = 10.000$ pa več kot 27%.

Algoritem FPB, ki ga bomo predstavili v nadaljevanju, vrača pakirano bazo le za grafe, ki imajo manj kot 115 točk, pri grafih z večjim številom točk pa lahko rezultat, ki ga vrne, uporabimo kot začetni približek pri računanju pakirane baze z algoritmom 3.10-I – ta bo s tako začetno bazo v nadaljevanju potreboval manj iteracij in bo zato deloval hitreje. Testi

so pokazali, da algoritem 3.10-I, ki začne z bazo $\{a_0, a_1\}$, v povprečju potrebuje $4, 4$ iteracij, če pa začne z bazo, ki jo vrne FPB, pa le $0, 9$ iteracij (v obeh primerih smo pakirano bazo računali za 10.000 naključno izbranih krožnih grafov s številom točk med 100 in 100.000).

Osnovna ideja algoritma FPB

Do zamisli za algoritem FPB smo prišli z empiričnim opazovanjem večjega števila krožnih grafov. V grobem jo lahko predstavimo takole: pakirano bazo modula X dobimo s projekcijami vektorjev a in b neke baze $\{a, b\}$ tega modula v smereh b in a. Natančneje, FPB pakirano bazo izračuna v naslednjih treh korakih:

- izračunaj začetno bazo $\{a, b\}$ modula X,
- naj bo a minimalni element traku $t(a, b)$ in
- naj bo b minimalni element traku $t(b, a)$.

Vhod tega algoritma so števila n, h_1 in h_2, izhod pa baza $\{a, b\}$.

Iskanje začetne baze za FPB

Že pri primerih, ki smo jih preverili ročno, smo ugotovili, da začetna baza ne sme biti povsem poljubna – izkazalo se je namreč, da baza, katere en vektor leži na ordinatni osi, ni dobro izhodišče za nadaljevanje. FPB mora torej v prvem koraku poiskati tako bazo $\{a, b\}$, da bo veljalo $a_2, b_2 \neq 0$. Pri iskanju take baze najprej upoštevamo dejstvo, da je determinanta bazne matrike modula X vedno enaka n. Ker vemo, da je en element množice X na ordinatni osi na višini $\frac{n}{\gcd(n,h_2)}$, lahko drug element te množice iščemo pri $x = \gcd(n, h_2)$ – bazni paralelogram baze $\{(0, \frac{n}{\gcd(n,h_2)}), (\gcd(n, h_2), t)\}$ ima ploščino n. Ker smo n, h_1 in h_2 izbrali tako, da je njihov največji skupni faktor enak 1, obstaja neskončno mnogo kandidatov za število t. Mi bomo t izbrali kot največjo nepozitivno rešitev enačbe

$$\gcd(n, h_2) * h_1 + t * h_2 \equiv 0 \pmod{n}. \tag{3.6}$$

Partikularno rešitev te enačbe dobimo z razširjenim Evklidovim algoritmom, ki pri vhodu (x, y) poleg $\gcd(x, y)$ vrne tudi tak par (s, t), da velja $s * x + t * y = \gcd(x, y)$ [35, 42] (predpostavili bomo naslednjo sintakso: funkcija ExtendedGCD(x,y) vrne rezultat oblike $\{\gcd(x, y), \{s, t\}\}$). Ker za vsak t, ki reši enačbo (3.6), obstaja tak s, da velja $\gcd(n, h_2) * h_1 + t * h_2 = s * n$, enačbo (3.6) rešimo tako, da najprej poiščemo $\{g, \{s', t'\}\} = \text{ExtendedGCD}(n, h_2)$. Potem velja: $s' * n + t' * h_2 = \gcd(n, h_2)$. Najmanjša nenegativna rešitev enačbe (3.6) je $t'' = \text{Mod}(-h_1 * t', n/\gcd(n, h_2))$, največja nepozitivna pa $t'' - n/\gcd(n, h_2)$ (z $\text{Mod}(x, y)$ označujemo ostanek po deljenju števila x z y).

Tako smo poiskali eno od baz za modul X in sicer, če z g označimo $g = \gcd(n, h_2)$,

$$(x, e) = \left\{ \left(0, \frac{n}{g}\right), \left(g, \text{Mod}(-h_1 * t', \frac{n}{g} - \frac{n}{g}\right) \right\},$$

ki pa ne izpolnjuje zahteve algoritma FPB, saj prvi vektor leži na ordinatni osi. Da bi se izognili tej težavi, vektor x zamenjamo z neko linearno kombinacijo vektorjev x in e, na primer z najmanjšim elementom, ki leži na traku $t(x,e)$, torej s $P(x,e)$. Trak $t(x,e)$ je strmo padajoč in vstopi v desno polravnino nad abscisno osjo (glej razdelek 3.9.1), zato je $P(x,e) = x + \lfloor -\frac{x_2}{e_2} \rceil e$. Tako smo prišli do začetne baze za algoritem FPB, ki se glasi

$$\{a,b\} = \left\{ e, x + \lfloor -\frac{x_2}{e_2} \rceil e \right\}.$$

Algoritem 3.10-II	FPB(n, h_1, h_2)
Ne-iterativni izračun pakirane baze za $n < 115$	

FPB(n, h_1, h_2) :

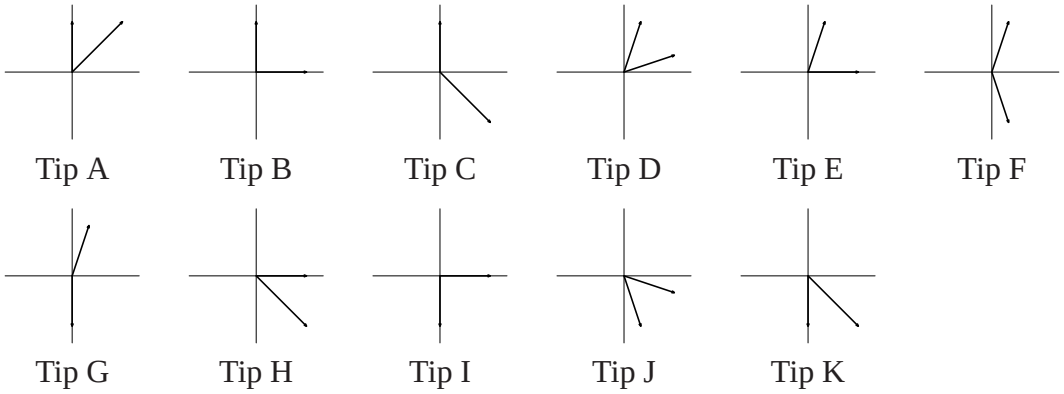
1: $\{g, \{s', t'\}\}$:=ExtendedGCD(n, h_2);
2: t :=Mod$(-t' * h_1, \frac{n}{g})$;
3: $x := (0, \frac{n}{g}),\ e = (g, t - \frac{n}{g})$;
4: $\{a,b\} := \left\{ e,\ x + \lfloor \frac{-x_2}{e_2} \rceil e \right\}$;
5: $a := P(a,b)$;
6: $b := P(b,a)$;
7: **return** $\{a,b\}$.

Modifikacija iterativnega algoritma

Če želimo algoritem FPB uporabiti za izračun začetne baze algoritma 3.10-I, je treba upoštevati, da algoritem 3.10-I na začetku izbere bazo v prvem in četrtem kvadrantu ter da ves čas skrbi, da so vrednosti pri x_i in y_i nenegativne. Algoritem 3.10-I popravimo tako, da prve štiri vrstice zamenjamo z:

1: $\{A, B\} = FPB(n, h_1, h_2)$;
2: **if** $A_1 < 0$ **then** $A := -A$;
if $B_1 < 0$ **then** $B := -B$;
3: $a_0 := \{x_0, y_0\} := \{A_1, |A_2|\}$;
4: $a_1 := \{x_1, -y_1\} := \{B_1, -|B_2|\}$.

Čeprav je tako spremenjeni algoritem hitrejši od prvotnega, saj v povprečju porabi bistveno manj iteracij (glej stran 57), se ocena časovne zahtevnosti za oba razlikuje le v konstantnem faktorju. Časovna zahtevnost najboljšega nam znanega algoritma za izračun pakirana baze, ki je prvi korak pri računanju premera, tako ostaja $\mathcal{O}(\log n)$. Obstoj eksplicitne formule za izračun premera neusmerjenih krožnih grafov (podoben rezultat je za usmerjene krožne grafe predstavljen v [8]) tako ostaja odprto vprašanje (glej poglavje *Nadaljevanje dela* ob koncu disertacije).



Slika 3.10-I: Shematski prikaz različnih tipov baze, glede na lego njenih vektorjev

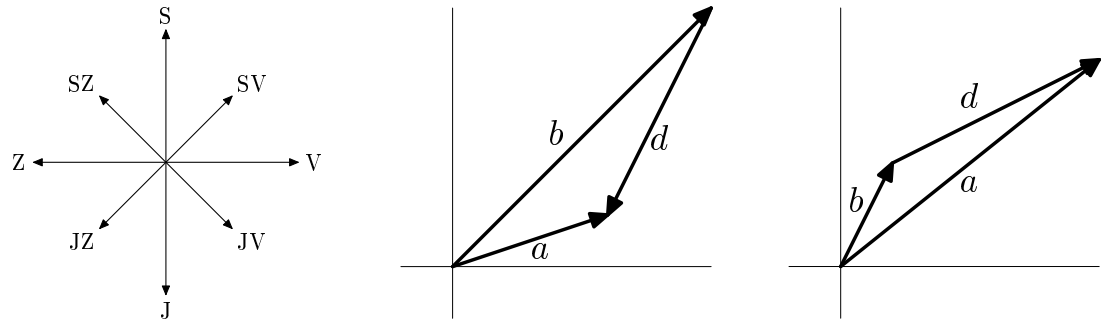
3.10.2 Tipizacija pakiranih baz

Glede na medsebojno lego baznih vektorjev bomo v nadaljevanju tega dela ločili 11 tipov pakiranih baz. Pri tem bomo predpostavili, da vektorja pakirane baze ležita bodisi v prvem bodisi v četrtem kvadrantu. Ta predpostavka zmanjša število posebnih primerov in je povsem realna, saj za bazo, ki jo vrne algoritem 3.10-I, to vedno velja, poleg tega pa lahko pakirano bazo, za katero to ne velja, s preprostimi operacijami (rotacija in zrcaljenje) preslikamo v pakirano bazo, ki ta pogoj izpolnjuje.

Bazo, na primer, katere oba vektorja ležita v notranjosti prvega kvadranta, imenujemo **baza tipa D**. Karakterizacija drugih tipov je razvidna iz tabele 3.10-I, shematsko pa so baze prikazane na sliki 3.10-I.

Tip baze	prvi vektor	drugi vektor
TIP A	prvi kvadrant, ordinatna os	prvi kvadrant, notranjost
TIP B	prvi kvadrant, ordinatna os	prvi kvadrant, abscisna os
TIP C	prvi kvadrant, ordinatna os	četrti kvadrant, notranjost
TIP D	prvi kvadrant, notranjost	prvi kvadrant, notranjost
TIP E	prvi kvadrant, notranjost	abscisna os
TIP F	prvi kvadrant, notranjost	četrti kvadrant, notranjost
TIP G	prvi kvadrant, notranjost	četrti kvadrant, ordinatna os
TIP H	abscisna os	četrti kvadrant, notranjost
TIP I	abscisna os	četrti kvadrant, ordinatna os
TIP J	četrti kvadrant, notranjost	četrti kvadrant, notranjost
TIP K	četrti kvadrant, notranjost	četrti kvadrant, ordinatna os

Tabela 3.10-I: Karakterizacija baz: tip baze je določen glede na lego baznih vektorjev



(a) (b) (c)

Slika 3.10-II: (a) Oznake za smer diagonale $d = a - b$, (b) lega vektorjev a in b v prvem kvadrantu pri diagonali d v smeri JZ in (c) lega vektorjev a in b v prvem kvadrantu pri diagonali d v smeri SV

3.10.3 Lastnosti pakirane baze

V nadaljevanju tega razdelka bomo predstavili nekatere lastnosti pakiranih baz, ki se nanašajo na medsebojno lego baznih vektorjev. Vektorja pakirane baze bomo označili z a in b in v vseh primerih predpostavili, da a in b ležita v prvem in/ali četrtem kvadrantu ter da vektor a leži "pod" vektorjem b (to je, da je kot med negativno ordinatno polosjo in vektorjem a manjši od kota med to polosjo in vektorjem b).

Smer diagonale
Najprej se bomo ustavili pri analizi smeri diagonale $d = a - b$. Pri poljubno izbranih vektorjih a in b bi lahko taka diagonala kazala v katerokoli od osmih smeri, kot prikazuje slika 3.10-II(a). Ker pa vektorja a in b nista poljubno izbrana, pač pa ju omejujejo določeni pogoji ($\{a, b\}$ je pakirana baza in a leži "pod" b), bomo pokazali, da večina od teh smeri ni mogoča. Oglejmo si najprej primere, ko bazna vektorja ležita v istem kvadrantu.

Trditev 3.10-II. *Če vektorja pakirane baze $\{a, b\}$ ležita v istem kvadrantu, je $a_1 \neq b_1$.*

Dokaz. Recimo, da velja $a_1 = b_1$. Gotovo je $b_1 \neq 0$, saj bi ob $a_1 = b_1 = 0$ vektorja a in b ne bila neodvisna. Ločiti moramo dva primera in sicer $a_2 < b_2$ in $a_2 > b_2$. V prvem velja:

$$||a - b|| = |a_1 - b_1| + |a_2 - b_2| = |b_2 - a_2| \leq |b_2| < ||b||.$$

Podobno pokažemo, da v drugem primeru velja $||a-b|| < ||a||$, kar pa se ob predpostavki o pakiranosti baze ne more zgoditi. —⋆— □

Trditev 3.10-III. *Za vektorja pakirane baze $\{a, b\}$ velja $a_2 \neq b_2$.*

Dokaz. Dokaz s protislovjem poteka podobno kot dokaz trditve 3.10-II. Predpostavimo, da velja $a_2 = b_2$ in ločimo dva primera: $a_1 > b_1$ in $a_1 < b_1$. V prvem, zaradi $a_2 \neq 0$, velja

$$||a-b|| = |a_1-b_1| + |a_2-b_2| = |a_1-b_1| \leq |a_1| < ||a||,$$

v drugem pa $||a-b|| < ||b||$. V obeh primerih torej pridemo do protislovja. —⋆— □

Trditvi 3.10-II in 3.10-III nam zagotavljata, da v primeru, ko a in b ležita v istem kvadrantu, nobena koordinata diagonale $d = a - b$ ne more biti nič (diagonala torej ni vektor v smeri abscisne ali ordinatne osi, oziroma smer diagonale ne more biti S, J, V ali Z). Pri trditvi 3.10-II je eksplicitna navedba pogoja, da vektorja ležita v istem kvadrantu, nujno potrebna, saj bomo v nadaljevanju videli, da ima lahko v primeru, ko ta ni izpolnjen, diagonala d smer J (glej trditev 3.10-VIII).

Trditev 3.10-IV. *Če pakirana baza $\{a, b\}$ leži v prvem kvadrantu, potem je diagonala $d = a - b$ vektor, ki leži v notranjosti četrtega kvadranta.*

Dokaz. Za vse smeri, razen za JV, pokažimo, da diagonala ne more biti tako usmerjena. Najprej vidimo, da smeri S, SZ in Z odpadejo, saj vektor a leži "pod" vektorjem b, nato s pomočjo trditev 3.10-II in 3.10-III izločimo še smeri J in V. Če bi imela diagonala d smer JZ, bi veljalo $||b|| = ||a|| + ||d|| \Rightarrow \ ||d|| < ||b||$ (glej sliko 3.10-II(b)), kar je v nasprotju s predpostavko o pakiranosti baze. Podobno protislovje bi dobili tudi v primeru smeri SV, saj bi veljalo $||a|| = ||b|| + ||d|| \Rightarrow \ ||d|| < ||a||$ (glej sliko 3.10-II(c)). Edina mogoča smer za vektor d je torej JV.

□

Neposredna posledica trditve 3.10-IV je, da ima diagonala $d = a - b$ baznih vektorjev prvega kvadranta smer JV, t.j. $d_1 > 0$ in $d_2 < 0$ (glej sliko 3.10-II(a)):

Posledica 3.10-V. *Za vektorja a in b pakirane baze, ki oba ležita v prvem kvadrantu, velja*

$$a_1 > b_1 \text{ in } a_2 < b_2.$$

Podobno kot smo za vektorja v prvem kvadrantu videli, da ima diagonala d smer JV, za vektorja v četrtem kvadrantu velja, da ima diagonala $d = a - b$ smer JZ.

Trditev 3.10-VI. *Če vektorja pakirane baze $\{a, b\}$ ležita v četrtem kvadrantu, potem je diagonala $d = a - b$ vektor, ki leži v notranjosti tretjega kvadranta.*

Dokaz. Dokaz poteka podobno kot pri trditvi 3.10-IV. Smeri V, SV in S izločimo zaradi predpostavke, da je a "pod" b, s smermi Z in J opravita trditvi 3.10-II in 3.10-III, za smeri SZ in JV pa pokažemo, da je $||d|| < ||b||$ oziroma $||d|| < ||a||$. Edina možnost za smer vektorja d je torej JZ. □

Posledica 3.10-VII. *Za vektorja a in b pakirane baze, ki oba ležita v četrtem kvadrantu, velja*

$$a_1 < b_1 \text{ in } |a_2| > |b_2|.$$

Za bazne vektorje, ki ležijo v istem kvadrantu, je torej smer vektorja d "do kvadranta" natančno določena. Drugače pa je, kadar vektorja a in b ležita vsak v svojem kvadrantu. Takrat kaže diagonalni vektor v eno od smeri JZ, J ali JV.

Trditev 3.10-VIII. *Če vektor a pakirane baze $\{a, b\}$ leži v četrtem, vektor b pa v prvem kvadrantu, potem za diagonalni vektor $d = a - b$ velja $d_2 < 0$.*

Dokaz. Smeri V, SV, S, SZ in Z niso mogoče, saj se ob predpostavki, da a leži v četrtem, b pa v prvem kvadrantu, ne pojavijo. Ostanejo torej samo smeri JZ, J in JV, pri vseh treh pa velja $d_2 < 0$, saj a leži pod b. □

Karakteristični koti vektorjev pakirane baze

Oglejmo si še nekaj lastnosti, ki se nanašajo na kote med vektorji pakirane baze ter na kote med posameznim vektorjem in koordinatnimi osmi.

Lema 3.10-IX. *Naj bo $\{a, b\}$ pakirana baza in naj bo a vektor četrtega, b pa vektor prvega kvadranta. Če z α označimo kot med absciso in vektorjem a ter z β kot med absciso in vektorjem b, potem velja bodisi $\alpha \leq 45°$, bodisi $\beta \leq 45°$.*

Dokaz. Recimo, da je $\alpha > 45°$ in $\beta > 45°$. Potem velja: $|a_2| > a_1$ in $b_2 > b_1$ (glej sliko 3.10-III(a)). Če upoštevamo definicijo in predznake komponent baznih vektorjev, dobimo $||a + b|| = ||(a_1 + b_1, a_2 + b_2)|| = a_1 + b_1 + |a_2 + b_2| = a_1 + b_1 + |b_2 - |a_2||$. Pri obravnavanju absolutne vrednosti $|b_2 - |a_2||$ bomo ločili dva primera in v obeh pokazali, da je diagonala $||a + b||$ krajša od ene od stranic, kar pa je v protislovju s predpostavko o pakiranosti baze.

(i) $b_2 \geq |a_2|$

$||a + b|| = a_1 + b_1 + b_2 - |a_2| = ||b|| + a_1 - |a_2| < ||b||$, saj je $a_1 - |a_2| < 0$. $\rightarrow\!\!\!\!\times\!\!\!\!\leftarrow$

(ii) $b_2 < |a_2|$

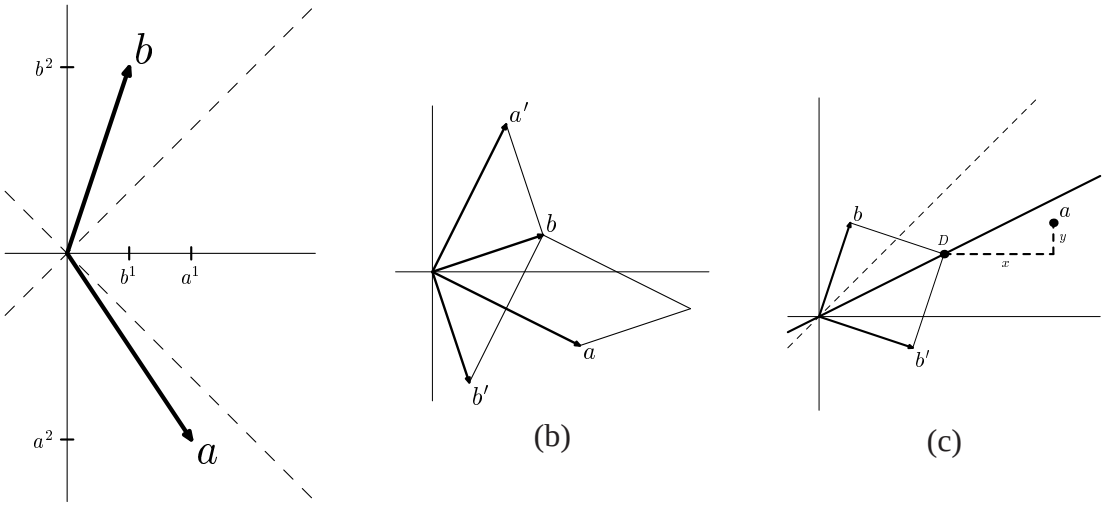
$||a + b|| = a_1 + b_1 + |a_2| - b_2 = ||a|| + (b_1 - b_2) < ||a||$, saj je $(b_1 - b_2) < 0$. $\rightarrow\!\!\!\!\times\!\!\!\!\leftarrow$

□

Neposredno iz leme 3.10-IX sledi, da kot med vektorjema pakirane baze, ki ležita vsak v svojem kvadrantu, ne more biti večji od 135°. Ker je kot med vektorjema istega kvadranta navzgor omejen z 90°, je 135° zgornja meja za kot med vektorjema pakirane baze v vseh primerih. Omenimo še, da je to natančna zgornja meja, saj obstajajo primeri, ko kot med vektorjema pakirane baze to vrednost doseže.

> Primer 3.10-I. Primer pakirane baze, pri kateri je kot med njenima vektorjema enak 135°, je baza $\{a, b\} = \{(3, -3), (0, 7)\}$ za graf $G(21;\ \pm 2, \pm 9)$.

Videli smo torej, da kot med vektorjema pakirane baze ne more biti poljubno velik, v nadaljevanju pa bomo pokazali, da tudi poljubno majhen ne more biti. Oglejmo si najprej naslednjo lemo.



Slika 3.10-III: Skice za pomoč pri dokazovanju leme 3.10-IX in trditve 3.10-XI

Lema 3.10-X. *Naj bo $\{a, b\}$ baza prostora X in naj vektorja a in b oba ležita v istem kvadrantu. Če z α (oz. β) označimo kot med vektorjem a (oz. b) in abscisno osjo in če je $\alpha < 45°$ in $\beta < 45°$, potem baza $\{a, b\}$ ni pakirana.*

Dokaz. Recimo, da vektorja a in b ležita v prvem kvadrantu in da sta kota med tema vektorjema in absciso manjša od 45°. Od tod sledi $a_1 > a_2$ in $b_1 > b_2$, iz posledice 3.10-V pa $a_1 > b_1$ in $a_2 < b_2$. Če z d označimo diagonalo $d = a - b$, dobimo protislovje: $||d|| = |a_1-b_1|+|a_2-b_2| = (a_1-b_1)+(b_2-a_2) = a_1-a_2-(b_1-b_2) \leq a_1+a_2-(b_1-b_2) < ||a||$. Pri tem prvi enačaj velja, ker $a_2 \geq 0$, drugi pa, ker je $(b_1 - b_2) > 0$. Podobno protislovje dobimo tudi v primeru, ko a in b ležita v četrtem kvadrantu. □

Na podoben način lahko pokažemo, da a in b ne moreta hkrati ležati v nobeni od štirih polovic prvega in četrtega kvadranta. Še več, velja celo, da ne glede na lego teh dveh vektorjev kot med njima ne more biti manjši od 45°. To je natančna spodnja meja, hitro pa najdemo primere, ki jo dosežejo.

> Primer 3.10-II. Primer pakirane baze, pri kateri je kot med njenima vektorjema enak 45°, je baza $\{a, b\} = \{(5, 0), (2, 2)\}$ za graf $G(10;\ \pm 2, \pm 3)$.

Trditev 3.10-XI. *Za kot γ med baznima vektorjema pakirane baze velja $\gamma \geq 45°$.*

Dokaz. Ločili bomo dva primera in sicer: (i) vektorja a in b ležita v prvem in četrtem kvadrantu in (ii) vektorja ležita v istem kvadrantu, in za vsakega posebej pokazali pravilnost leme. 65

(i) Naj bo $\{a, b\}$ pakirana baza, vektor a naj leži v četrtem, vektor b pa v prvem kvadrantu in naj bo kot med a in b manjši ali enak 45°. Z $a' = (-a_2, a_1)$ označimo pravokotni

komplement vektorja a, z $b' = (b_2, -b_1)$ pa pravokotni komplement vektorja b (glej sliko 3.10-III(b)). Velja $||a|| = ||a'||$ in $||b|| = ||b'||$. Če z d_1 in d_2 označimo obe diagonali paralelograma $[a, b)$ ($d_1 = a - b$ in $d_2 = a + b$), z d_1' in d_2' pa diagonali paralelograma $[a', b')$, velja

$$||d_1'|| = | - a_2 - b_2| + |a_1 + b_1| = ||d_2||$$

in

$$||d_2'|| = | - a_2 + b_2| + |a_1 - b_1| = ||d_1||.$$

Ker je $\{a, b\}$ pakirana baza, je $\min(||a||, ||b||) \leq \min(||d_1||, ||d_2||)$, od tod pa sledi tudi, da je

$$\min(||a'||, ||b'||) \leq \min(||d_1'||, ||d_2'||), \tag{3.7}$$

saj so paroma enake norme leve in desne strani obeh neenačb. Ker je kot med a in b manjši ali enak 45°in ker a in b ne ležita na abscisni osi, sta tako kot med absciso in vektorjem a' kot kot med absciso in vektorjem b' večja od 45°. Zato po lemi 3.10-IX baza $\{a', b'\}$ ni pakirana, kar pa je v nasprotju z neenakostjo 3.7, torej v protislovju s predpostavko o pakiranosti baze. $\multimap\!\!\!\!\times\!\!\!\!\multimap$

(ii) Recimo, da oba bazna vektorja pakirane baze $\{a, b\}$ ležita v prvem kvadrantu (dokaz za četrti kvadrant poteka simetrično). Naj bo $b' = (b_2, -b_1)$ pravokotni komplement vektorja b. Z $D = b + b' = (b_1 + b_2, b_2 - b_1)$ označimo četrto oglišče paralelograma $[b, b']$, s p_D pa premico, ki poteka skozi točki 0 in D (glej sliko 3.10-III(c)). Ker po lemi 3.10-X vektor b ne more ležati pod simetralo kvadranta, velja $b_1 \leq b_2$ in zato $||D|| = 2b_2$. Dokažimo sedaj, da vektor a neodvisno od svoje horizontalne lege glede na vektor D (t.j. $a_1 < D_1$, $a_1 = D_1$ ali $a_1 > D_1$) vedno leži na ali pod premico p_D. Od tod namreč neposredno sledi, da je kot med a in b vsaj 45°.

a) Vektor a, ki leži levo od D, je vsaj toliko pomaknjen tudi pod D, saj bi bila v nasprotnem primeru dolžina diagonale $a - b$ krajša od b in baza ne bi bila pakirana. Ker ima premica p_D naklon manjši ali kvečjemu enak 45°, točka a, ki je levo od D, torej leži na ali pod p_D.

b) Naj a leži na vertikali nad točko D, $a = (D_1, D_2 + x) = (b_1 + b_2, b_2 - b_1 + x)$. Po trditvi 3.10-IV ima diagonala $a - b$ smer JV in velja (glej posledico 3.10-V) $b_2 - b_1 + x < b_2$, oziroma $x < b_1$. Za velikost diagonale potem velja: $||a - b|| = |b_2| + |x - b_1| < |b_2| + |b_1| = ||b||$, kar pa ne more biti, saj je baza pakirana. Točka a torej, ki leži na vertikali skozi točko D, ne more ležati nad premico p_D.

c) Naj a leži desno od D in naj bo $a = D + (x, y)$ pri $x > 0$. V primeru, da je $y \leq 0$, a leži pod premico p_D. Oglejmo si torej primer, ko $y > 0$. Ker so vse koordinate vektorjev D in (x, y) pozitivne, velja $||a|| = ||D|| + x + y$ in $||a - b|| = ||D|| + x - y = ||b|| + x - y$ (glej sliko 3.10-III(c)). Iz tega pa neposredno sledi $||a|| = ||D|| + x + y = 2b_2 + x + y > 2b_2 + x - y \geq ||b|| + x - y = ||a - b||$, kar pa je v nasprotju s predpostavko o pakiranosti baze.

⋮
63 ⌊ □

Povzetek
Povzemimo na kratko najpomembnejše lastnosti pakirane baze, ki smo jih predstavili v tem razdelku. Tudi tu predpostavljamo, da vektorja pakirane baze $\{a, b\}$ ležita v prvem in/ali četrtem kvadrantu ter da vektor a leži "pod" vektorjem b. Z d označimo diagonalo $d = a - b$.

- ⇝ za elementa pakirane baze velja: $\max\{||a||, ||b||\} \leq \min\{||a + b||, ||a - b||\}$,
- ⇝ če vektorja pakirane baze ležita v istem kvadrantu, je $a_1 \neq b_1$ in $a_2 \neq b_2$,
- ⇝ če vektorja pakirane baze ležita v prvem kvadrantu, ima diagonala d smer JV in velja $a_1 > b_1$ in $a_2 < b_2$,
- ⇝ če vektorja pakirane baze ležita v četrtem kvadrantu, ima diagonala d smer JZ in velja $a_1 < b_1$ in $|a_2| > |b_2|$,
- ⇝ če en vektor leži v prvem, drugi pa v četrtem kvadrantu, potem ima diagonala d smer JZ, J ali JV,
- ⇝ diagonala d pakirane baze nikoli ni usmerjena v smer V, Z, S, SV ali SZ,
- ⇝ kot med vektorjema pakirane baze je večji ali enak 45° in manjši ali enak 135°; obe meji sta natančni,
- ⇝ če vektorja pakirane baze ležita vsak v svojem kvadrantu, potem vsaj za enega od njiju velja, da je kot med njim in abscisno osjo manjši od 45°.

3.11 Premer in najkrajše poti v dvokrožnih grafih

V tem razdelku bomo predstavili algoritme za računanje razdalje med točkami, premera grafa in najkrajših poti med parom točk grafa $G(n;\ \pm h_1, \pm h_2)$. Kot smo videli v razdelku 3.5, lahko na sprehod v dvokrožnem grafu gledamo iz različnih zornih kotov. Predstavimo ga lahko kot zaporedje povezav, kot rešitev kongruenčne enačbe ali kot element mreže $\mathbb{Z}_2^2$. V tem razdelku bomo pri predstavitvi algoritmov uporabili slednjega, na koncu razdelka pa bomo na dobljene rešitve pogledali še kot na rešitve kongruenčnih enačb.

3.11.1 Dolžina najkrajše poti

Označimo z d_w razdaljo med točkama 0 in w grafa $G(n;\ \pm h_1, \pm h_2)$, to je dolžino najkrajše poti med točkama 0 in w, in poglejmo, kako bi to količino izračunali s pomočjo mreže $\mathbb{Z}_2^2$. Definirajmo najprej (na standardni način) razdaljo med točko in množico v $\mathbb{Z}_2^2$.

Definicija 3.11-I. Razdalja točke $x \in \mathbb{Z}_2^2$ od množice $Z \subset \mathbb{Z}_2^2$ *je najkrajša razdalja med kakim elementom množice* Z *in točko x, oziroma*

$$d(x, Z) := \min_{z \in Z} d(x, z).$$

Podobno definiramo razdaljo med dvema množicama kot najmanjšo razdaljo med poljubnima dvema predstavnikoma teh dveh množic. Pri računanju dolžine d_w si bomo pomagali z lastnostjo, da je razdalja med množicama X in X_w enaka razdalji med poljubnim predstavnikom prve množice in drugo množico. Da to res drži, se prepričamo z naslednjo trditvijo.

Trditev 3.11-II. *Naj bo* $w \in V_n$. *Za poljubna* x_1, x_2 *iz* X_w *velja* $d(x_1, X) = d(x_2, X)$.

Dokaz. Najprej dokažimo, da je $d(x_1, X) \leq d(x_2, X)$. Naj bo d razdalja med x_2 in X, $d := d(x_2, X)$. Potem obstaja tak $x \in X$, da je $d(x_2, x) = d$. Dokazali bi radi, da obstaja element $x' \in X$, ki je od x_1 oddaljen natanko d. Zaradi simetričnosti celotnega problema takega elementa ni težko poiskati – če x premaknemo za $x_1 - x_2$, dobimo element x', ki je od x_1 oddaljen enako, kot je x oddaljen od x_2. Zapišimo to še formalno.
Naj bo $x' = x + (x_1 - x_2)$. Velja: $l(x') = l(x + x_1 - x_2) = l(x) + l(x_1) + l(x_2) = l(x) = 0$, zato $x' \in X$ in velja:

$$d(x_1, X) \leq d(x_1, x') = ||x_1 - x'|| = ||x_1 - x - x_1 + x_2|| = d(x, x_2) = d.$$

Od tod sledi: $d(x_1, X) \leq d(x_2, X)$. Podobno dokažemo tudi obratno, t.j. $d(x_1, X) \geq d(x_2, X)$, in s tem je trditev dokazana. □

Ker X_w predstavlja vse (regularne) sprehode med točkama 0 in w grafa $G(n;\ \pm h_1, \pm h_2)$, je razdalja d_w enaka najmanjši normi elementov v X_w, oziroma

$$d_w = \min_{x_w \in X_w} ||x_w||.$$

S pomočjo tega dejstva dokažemo naslednjo trditev.

Trditev 3.11-III. *Razdalja* d_w *je enaka razdalji med množicama* X_w *in* X, *to je*

$$d_w = d(X_w, X).$$

Dokaz.

$$\begin{aligned} d(X_w, X) &= \min_{x \in X} \min_{x_w \in X_w} d(x_w, x) = \min_{x \in X} \min_{x_w \in X_w} d(x_w - x, 0) \overset{*}{=} \min_{x \in X} \min_{x_w \in X_w} d(x_w, 0) = \\ &= \min_{x \in X} \min_{x_w \in X_w} ||x_w|| = \min_{x \in X} d_w = d_w \end{aligned}$$

Ker je $l(x) = 0$ za vsak $x \in X$, je $w - x \in X_w$ in enačaj * res velja. □

Če združimo trditvi 3.11-II in 3.11-III, dobimo

$$d_w \overset{trd.3.11-III}{=} d(X_w, X) \overset{trd.3.11-II}{=} d(x_w, X) \text{ za } \forall x_w \in X_w.$$

Pravkar napisana formula kar sama od sebe ponuja naslednje navodilo: d_w izračunaš tako, da vzameš poljubnega predstavnika množice X_w in izračunaš njegovo oddaljenost od množice X. V splošnem primeru, kjer je računanje take oddaljenosti zapletena naloga, s formulo ne pridobimo ničesar, v našem primeru pa, ko vsa pripravljena orodja omogočajo izračun te razdalje v konstantnem času, formula pomeni veliko pridobitev. Trditev 3.11-IV, ki jo bomo predstavili v nadaljevanju, nam zagotavlja, da lahko v primeru, ko za bazo modula X izberemo pakirano bazo in za predstavnika množice X_w osnovni bazni element x_w, razdaljo med x_w in X izračunamo v konstantnem času.

Trditev 3.11-IV. *Naj bo $\{a, b\}$ pakirana baza množice X in w poljubna oznaka v mreži $\mathbb{Z}_2^2$ ($w \in V_n$). Razdalja med osnovnim baznim elementom $x_w \in X_w \cap [a, b)$ in množico X je dosežena v ogliščih paralelograma $[a, b)$, oziroma*

$$d(x_w, X) = \min\{||x_w||, ||x_w - a||, ||x_w - b||, ||x_w - (a + b)||\}.$$

Trditev 3.11-IV sta dokazala Žerovnik in Pisanski v [54], zato dokaza tu ne bomo ponavljali. Poudarili bomo le, da je pogoj za pakiranost baze potreben – brez tega pogoja trditev ne drži, v kar se prepričamo v primeru 3.11-I.

> Primer 3.11-I. Na sliki 3.6-II(a) na strani 39 je prikazana baza, ki ni pakirana. Razdalja osnovnega baznega elementa $x_4 = (1, -2)$ do štirih oglišč baznega paralelograma je 3, razdalja med X_4 in X pa je enaka 2 (najbližja "ničla" je element (-1,-2)). Trditev 3.11-IV v primeru, da baza ni pakirana, ne drži.

Algoritem 3.11-I za izračun dolžine najkrajše poti med točkama 0 in w grafa $G(n;\ \pm h_1, \pm h_2)$ temelji na trditvi 3.11-IV, njegova časovna zahtevnost pa je $\mathcal{O}(\log n)$. Prva dva koraka potrebujeta vsak po $\mathcal{O}(\log n)$ časa (glej razdelka 3.10 in 3.8), zadnji korak pa se izvrši v konstantnem času.

Algoritem 3.11-I	$d_w(n, h_1, h_2)$
Računanje razdalje d_w med točkama 0 in w v $G(n;\ \pm h_1, \pm h_2)$	

$d_w(n, h_1, h_2)$:

1: $\{a, b\}$:= pakirana baza modula X v $G(n;\ \pm h_1, \pm h_2)$;

2: x_w:= osnovni bazni element;

3: **return** $\min\{||x_w||, ||x_w - a||, ||x_w - b||, ||x_w - (a + b)||\}$.

3.11.2 Premer

Z uporabo trditve 3.11-IV lahko hitro izračunamo tudi premer grafa $G(n;\ \pm h_1, \pm h_2)$, ki ga označimo z diam($G(n;\ \pm h_1, \pm h_2)$) in definiramo z

$$\text{diam}(G(n;\ \pm h_1, \pm h_2)) := max_{u,v \in V_n} d(u, v).$$

Ker je v krožnem grafu razdalja med poljubnima točkama u in v enaka razdalji med točkama 0 in $w = v - u \pmod n$, premer krožnega grafa izračunamo tako, da izračunamo največjo razdaljo med točko 0 in drugimi točkami, oziroma, po trditvi 3.11-IV, da izračunamo največjo razdaljo med oglišči in drugimi točkami osnovnega baznega paralelograma. Ker je v paralelogramu od oglišč najbolj oddaljena sredinska točka (točka, ki leži na sečišču diagonal), je računanje premera ekvivalentno računanju "velikosti" paralelograma, t.j. oddaljenosti sečišča diagonal od oglišč. Pri tem moramo seveda upoštevati, da sečišče diagonal morda ni element $\mathbb{Z}_2^2$, in je zato treba poiskati najbližji celoštevilčni element. Pri dokazovanju pravilnosti algoritma 3.11-II za izračun premera dvokrožnega grafa, ki svoje delo opravi v $\mathcal{O}(\log n)$ časa, potrebujemo naslednjo lemo [54].

Lema 3.11-V. *Naj bo $\{a, b\}$ pakirana baza modula X za graf $G(n;\ \pm h_1, \pm h_2)$ in naj bosta d_1 in d_2 dolžini krajše in daljše diagonale baznega paralelograma. Če je $d_1 = d_2 > \max(||a||, ||b||)$ in sta obe normi $||a||$ in $||b||$ lihi, potem je diam($G(n;\ \pm h_1, \pm h_2)$)=$\lfloor d_1/2 \rfloor - 1$, sicer pa diam($G(n;\ \pm h_1, \pm h_2)$)=$\lfloor d_1/2 \rfloor$.*

Algoritem 3.11-II	diam($G(n;\ \pm h_1, \pm h_2)$)
Računanje premera grafa $G(n;\ \pm h_1, \pm h_2)$	

diam($G(n;\ \pm h_1, \pm h_2)$) :

1: $\{a, b\}$:= pakirana baza modula X;
2: $d_1 := \min(||a + b||, ||a - b||)$;
3: $d_2 := \max(||a + b||, ||a - b||)$;
4: $w := \lfloor d_1/2 \rfloor$;
5: **if** $(d_1 = d_2 > \max(||a||, ||b||))$ **and** ($||a||$ in $||b||$ liha) **then** $w = w - 1$;
6: **return** w.

3.11.3 Najkrajša pot

Doslej smo s pomočjo lastnosti pakirane baze izračunali razdaljo med parom točk in premer grafa. Na enako preprost način lahko izračunamo tudi najkrajšo pot med danim parom točk. Trditev 3.11-IV, ki zagotavlja, da je razdalja med točkama 0 in w enaka razdalji med osnovnim baznim elementom x_w in štirimi oglišči osnovnega baznega paralelograma

$O = \{o_1, o_2, o_3, o_4\}$, lahko razumemo tudi širše. Če je namreč $o \in O$ tisto oglišče baznega paralelograma, ki minimizira razdaljo do točke x_w ($d(x_w, o) = \min_{v \in O} d(x_w, v)$), potem predstavlja element $x_w - o$ najkrajšo pot med točkama 0 in w. Res, saj velja $x_w - o \in X_w$ in $||x_w - o|| = d(x_w, o) = \min_{v \in O} d(x_w, v)$. Algoritem za iskanje najkrajše poti mora torej med štirimi kandidati množice $\{x_w - o_1, x_w - o_2, x_w - o_3, x_w - o_4\}$ poiskati tistega z najmanjšo normo [55]. Dejstvo, da je pravkar omenjena množica enaka množici baznih elementov za množico X_w, potrjuje pravilnost algoritma 3.11-III.

Algoritem 3.11-III	$p_w(n, h_1, h_2)$
Računanje najkrajše poti p_w med 0 in w v $G(n;\ \pm h_1, \pm h_2)$	

$p_w(n, h_1, h_2)$:

1: $\{a, b\} :=$ pakirana baza modula X v $G(n;\ \pm h_1, \pm h_2)$;

2: $C_1^w :=$ množica baznih elementov;

3: **return** $\min_{\ell_1} C_1^w$.

Algoritem 3.11-III pakirano bazo poišče s pomočjo algoritma 3.10-I v $\mathcal{O}(\log n)$, množico C_1^w s pomočjo postopka za izračun osnovnega baznega elementa, ki je opisan na strani 49, v $\mathcal{O}(k^3)$, najmanjši element množice C_1^w pa v konstantnem času. Skupna časovna zahtevnost algoritma 3.11-III je torej $\mathcal{O}(\log n)$. Spomnimo še na dejstvo, da obstajajo primeri, ko najkrajša pot ni ena sama – v tem primeru algoritem vrne eno od njih (katero, je odvisno od implementacije).

Rešitev kongruenčne enačbe z enim dodatnim pogojem

V poglavju 3.5 smo videli, da je iskanje najkrajše poti med točkama 0 in w dvokrožnega grafa $G(n;\ \pm h_1, \pm h_2)$ ekvivalentno iskanju take rešitve enačbe

$$x_1 h_1 + x_2 h_2 \equiv w \pmod{n}, \tag{3.8}$$

ki minimizira vsoto

$$|x_1| + |x_2|. \tag{3.9}$$

S predstavitvijo algoritma 3.11-III za izračun najkrajše poti smo torej hkrati predstavili tudi algoritem za reševanje kongruenčne enačbe 3.8 ob pogoju 3.9. Algoritem ima časovno zahtevnost $\mathcal{O}(\log n)$.

3.12 Delno usmerjeni krožni grafi in omejene poti

Med razvojem usmerjevalnih algoritmov za dvokrožne grafe se je pokazala potreba po poznavanju lastnosti neke druge (dvokrožnim grafom sorodne) družine grafov. Grafe te družine, ki smo jih poimenovali *delno usmerjeni dvokrožni grafi*, bomo predstavili v tem

razdelku, v naslednjem pa bomo predstavili algoritem za računanje najkrajših poti med njihovimi točkami. V poglavju 4 bomo nato pokazali, kako lahko z uporabo tega algoritma izvedemo hitro usmerjanje v dvokrožnih grafih.

Delno usmerjeni grafi

Pri predstavitvi nove strukture bomo izhajali iz definicije KGC na strani 24, s katero smo množico povezav neusmerjenih dvokrožnih grafov definirali kot unijo množic E_i pri $i \in I$ (množice E_i vsebujejo povezave tipa $\mathcal{E}_i$). Z izrazom *delno usmerjen dvokrožni graf* bomo označevali grafe, ki so v vsem, razen v številu povezav, enaki neusmerjenim dvokrožnim grafom. Množico povezav delno usmerjenih grafov dobimo tako, da iz množice povezav neusmerjenih grafov odstranimo vse povezave enega od štirih tipov. Vsaka točka v delno usmerjenem grafu je z grafom povezana s tremi povezavami. Če smo na neusmerjen graf gledali kot na graf z neusmerjenimi kratkimi in neusmerjenimi dolgimi povezavami, lahko na delno usmerjen graf gledamo kot na graf, v katerem so ene od teh povezav (dolge oziroma kratke) usmerjene, druge pa neusmerjene. Delno usmerjene grafe bomo definirali splošno (za poljuben k), čeprav jih bomo v nadaljevanju uporabljali le v primeru $k = 2$. V spodnji definiciji uporabljamo splošne oznake, privzete v razdelku 3.2.

Definicija 3.12-I. *Za $i \in I$ označimo z E_i množico povezav tipa $\mathcal{E}_i$, to je $E_i = \{(v, v + \text{sign(i)} * h_{|i|} \pmod n;\ v \in V_n\}$. Graf z množico točk V_n in množico povezav $E = \cup_{j \in I - \{i\}} E_j$ imenujemo* delno usmerjen k-krožni graf *in označimo z $G^i(n;\ \pm h_1, \ldots, \pm h_k)$.*

> Primer 3.12-I. V delno usmerjenih dvokrožnih grafih ima vsaka točka tri sosede. Tako je, na primer, v delno usmerjenem dvokrožnem grafu $G^1(8;\ \pm 2, \pm 3)$ vsaka točka u povezana s točkami $u - 2 \pmod 8$ ter $u \pm 3 \pmod 8$. Povezave tipa $u + 2 \pmod 8$ v tem grafu ne obstajajo.

Omejene najkrajše poti

Podobno kot pri usmerjenih in neusmerjenih, lahko tudi pri delno usmerjenih dvokrožnih grafih vsak regularen sprehod med dvema točkama predstavimo s parom števil (x_1, x_2), kjer x_1 predstavlja število kratkih (t.j. v smeri h_1 oziroma $-h_1$), x_2 pa število dolgih (t.j. v smeri h_2 oziroma $-h_2$) povezav. Predznak števil x_1 in x_2 pove smer povezave: pozitivna števila predstavljajo povezave v smeri naprej, negativna pa povezave v smeri nazaj. Formalno: naj bo $i \in I^+$ in $j =$ sign(x_i) $* i$, potem x_i v paru (x_1, x_2) predstavlja $|x_i|$ povezav tipa e_j.

Naj bo (x_1, x_2) poljuben regularen sprehod v delno usmerjenem dvokrožnem grafu $G^i(n;\ \pm h_1, \pm h_2)$. Prepoved uporabe povezave tipa e_i se odraža tudi v vrednosti števil x_1 in x_2 – zaloga vrednosti za eno od njiju (konkretno: za $x_{|i|}$) je omejena. Medtem ko lahko števili, ki predstavljata sprehod v neusmerjenem grafu, zavzameta celoštevilsko vrednost z intervala $(-n, \ldots, n)$, lahko v delno usmerjenem primeru število $x_{|i|}$ zavzame le polovico teh vrednosti (samo pozitivne ali samo negativne).

> Primer 3.12-II. Za vsak regularen sprehod (x_1, x_2) v grafu $G^1(n;\ \pm h_1, \pm h_2)$ velja $x_1 \leq 0$, saj $x_1 > 0$ predstavlja povezave tipa e_1, ki pa so v tem primeru prepovedane.

Za regularen sprehod v delno usmerjenem dvokrožnem grafu $G^i(n;\ \pm h_1, \pm h_2)$, ki je predstavljen s parom (x_1, x_2), torej v splošnem velja:

$$\text{sign}(i) * x_{|i|} \leq 0 \qquad \text{in} \qquad x_{\hat{i}} \in \mathbb{Z}.$$

Z $\hat{i}$ smo označili "neproblematični" tip povezav (torej, za $i = 1$ in $i = -1$ je $\hat{i} = 2$, za $i = 2$ in $i = -2$ pa je $\hat{i} = 1$), oziroma

$$\hat{i} = (|i| \ mod\ 2) + 1.$$

Po omejenosti vrednosti, ki jih lahko zavzameta števili x_1 in x_2, se imenuje tudi najkrajša pot v delno usmerjenem grafu.

Definicija 3.12-II. *Najkrajšo pot v delno usmerjenem dvokrožnem grafu $G^i(n;\ \pm h_1, \pm h_2)$ med točkama u in v imenujemo* omejena najkrajša pot tipa **i**, *označimo pa jo s $p_i^{(u,v)}$; njeno dolžino označimo z $d_i^{(u,v)}$.*

Sprehod $p_i^{(u,v)}$ med točkama u in v delno usmerjenega dvokrožnega grafa je hkrati tudi sprehod med enako imenovanima točkama v neusmerjenem grafu $G(n;\ \pm h_1, \pm h_2)$ (čeprav morda ne najkrajši).

> Primer 3.12-III. V grafu $G(8;\ \pm 2, \pm 3)$ predstavlja pot $0 \xrightarrow{+2} 2$ najkrajšo pot med točkama 0 in 2. Ta pot je hkrati enaka omejenim najkrajšim potem $p_i^{(0,2)}$ za vsak $i \neq 1$. Omejena najkrajša pot pri $i = 1$, torej pot, vzdolž katere so povezave tipa $\mathcal{E}_1$ prepovedane, pa je $p_1^{(0,2)}$: $0 \xrightarrow{-3} 5 \xrightarrow{-3} 2$.

Dolžina omejenih najkrajših poti

V delno usmerjenem dvokrožnem grafu za vsak par točk u in v obstajajo štiri (ne nujno različne) omejene najkrajše poti $p_{-2}^{(u,v)}$, $p_{-1}^{(u,v)}$, $p_1^{(u,v)}$ in $p_2^{(u,v)}$. Označimo z $d^{(u,v)}$ dolžino najkrajše, z $D^{(u,v)}$ pa dolžino najdaljše med njimi:

$$d^{(u,v)} := \min\{d_i^{(u,v)}; i \in I\}, \quad D^{(u,v)} := \max\{d_i^{(u,v)}; i \in I\}.$$

Naj bodo i, j, k, l med seboj različni elementi množice I (t.j. $\{i, j, k, l\} = I$). Potem glede na vrednost števil $d^{(u,v)}$ in $D^{(u,v)}$ razlikujemo dva primera. V prvem, to je pri $d^{(u,v)} = D^{(u,v)}$, so vse omejene najkrajše poti hkrati tudi najkrajše v neusmerjenem grafu. Število različnih omejenih najkrajših poti je število ekvivalenčnih razredov poti med danim parom točk. Običajno sta taka razreda dva, so pa tudi primeri, ko jih je več.

> Primer 3.12-IV. V grafu $G(10;\ \pm 1, \pm 3)$ so omejene najkrajše poti med točkama 0 in 5 enake $p_{-2}^{(0,5)} = (-1, 2)$, $p_{-1}^{(0,5)} = (2, 1)$, $p_1^{(0,5)} = (-1, 2)$ in $p_2^{(0,5)} = (1, -2)$. V tem grafu torej obstajajo vsaj trije različni ekvivalenčni razredi poti med točkama 0 in 5 in sicer: [-1,2], [2,1] in [1,-2]. Ker je tudi [-2,-1] ekvivalenčni razred poti med 0 in 5, imamo v tem grafu štiri različne ekvivalenčne razrede najkrajših poti med točkama 0 in 5.

V drugem primeru, to je pri $d^{(u,v)} \neq D^{(u,v)}$, sta dve izmed števil $d_i^{(u,v)}$ enaki, v kar se prepričamo z naslednjo lemo.

Lema 3.12-III. *Če je $d^{(u,v)} \neq D^{(u,v)}$, potem obstajata taka $i, j \in I$, da velja $i \neq j$ in $d_i^{(u,v)} = d_j^{(u,v)} = d^{(u,v)}$.*

Dokaz. Po definiciji števila $d^{(u,v)}$ gotovo obstaja tak $i \in I$, da je $d_i^{(u,v)} = d^{(u,v)}$. Ker je $d_i^{(u,v)}$ regularna pot in zato vsebuje kvečjemu dva različna tipa povezav, obstaja tak $j \in I$, da $j \neq i$ in $\mathcal{E}_j$ ni del poti $p_i^{(u,v)}$. Najkrajša pot brez povezave tipa $\mathcal{E}_j$ torej ne more biti daljša od $p_i^{(u,v)}$, krajša pa tudi ne, saj je $p_i^{(u,v)}$ najkrajša med vsemi omejenimi potmi. Zato velja $d_j^{(u,v)} = d^{(u,v)}$. □

Glede na velikost števil $d_i^{(u,v)}$ v primeru $d^{(u,v)} \neq D^{(u,v)}$ velja bodisi $d_i = d_j < d_k = d_l$ bodisi $d_i = d_j = d_k < d_l$ bodisi $d_i = d_j < d_k < d_l$.

Primer 3.12-V. Na primeru grafa $G(12;\ \pm 4, \pm 5)$ se prepričajmo, da obstajajo vse tri možnosti glede velikosti števil $d_i^{(u,v)}$.

[1.] $p_{-2}^{(0,1)} = (-1, 1)$, $p_{-1}^{(0,1)} = (2, 1)$, $p_1^{(0,1)} = (-1, 1)$ in $p_2^{(0,1)} = (1, -3)$, zato

$$d_{-2}^{(0,1)} = d_1^{(0,1)} < d_{-1}^{(0,1)} < d_2^{(0,1)}.$$

[2.] $p_{-2}^{(0,2)} = (1, 2)$, $p_{-1}^{(0,2)} = (0, -2)$, $p_1^{(0,2)} = (0, -2)$ in $p_2^{(0,2)} = (0, -2)$, zato

$$d_{-1}^{(0,2)} = d_1^{(0,2)} = d_2^{(0,2)} < d_{-2}^{(0,2)}.$$

[3.] $p_{-2}^{(0,3)} = (0, 3)$, $p_{-1}^{(0,3)} = (0, 3)$, $p_1^{(0,3)} = (-1, -1)$ in $p_2^{(0,3)} = (-1, -1)$, zato

$$d_1^{(0,3)} = d_2^{(0,3)} < d_{-1}^{(0,3)} = d_{-2}^{(0,3)}.$$

S sistematičnim pregledovanjem vseh dvokrožnih grafov s številom točk med 5 in 100 (teh grafov je približno 55.000) smo ugotovili, da sta v 4,4% vseh primerov števili $d^{(u,v)}$ in $D^{(u,v)}$ enaki, v preostalih 95,6% pa različni. Pri $d^{(u,v)} = D^{(u,v)}$ smo v 94,8% našteli dva ekvivalenčna razreda, v preostalih 5,2% pa tri oziroma štiri. Pri $d^{(u,v)} \neq D^{(u,v)}$ smo ugotovili, da se primer $d_i = d_j < d_k = d_l$ pojavi v 23,3%, primer $d_i = d_j < d_k < d_l$ v 47,5%, primer $d_i = d_j = d_k < d_l$ pa v preostalih 29,2% primerih.

Prevedba problema

Podobno kot v usmerjenem in neusmerjenem bomo tudi v omejenem primeru poiskali ekvivalent sprehoda v celoštevilski mreži $\mathbb{Z}_2^2$. Točka x označene celoštevilske mreže $\mathbb{Z}_2^2$ z oznako w bo predstavljala ekvivalenčni razred sprehodov med točkama u in v, za katere je $v - u = w \pmod n$. Zaradi omejenosti sprehodov bomo morali omejiti tudi velikost mreže. Ko bomo obravnavali graf $G^i(n;\ \pm h_1, \pm h_2)$, bodo namreč zanimive samo tiste točke (x_1, x_2) mreže $\mathbb{Z}_2^2$, za katere velja $\text{sign}(i) * x_{|i|} \leq 0$, torej točke polravnine $\mathbb{Z}_{-i}^2$. Pri iskanju omejenih najkrajših poti bomo namesto množice X_w pregledovali množico $X_w \cap \mathbb{Z}_{-i}^2$.

3.13 Omejene najkrajše poti v dvokrožnih grafih

Problem računanja najkrajših poti v delno usmerjenem dvokrožnem grafu $G^i(n;\ \pm h_1, \pm h_2)$ lahko predstavimo na več načinov.

A: Pot v grafu. Izračunaj najkrajšo pot $p_i^{(u,v)}$ med danim parom točk (u, v) delno usmerjenega dvokrožnega grafa $G^i(n;\ \pm h_1, \pm h_2)$.

B: Kongruenčna enačba. Za dani vrednosti $u, v \in V_n$ poišči tako rešitev (x_1, x_2) enačbe

$$u + x_1 h_1 + x_2 h_2 \equiv v \pmod{n}, \tag{3.10}$$

ki med vsemi pari $(x_1, x_2) \in \mathbb{Z}_2^2$, za katere velja $\text{sign}(i) * x_{|i|} \leq 0$, minimizira $|x_1| + |x_2|$.

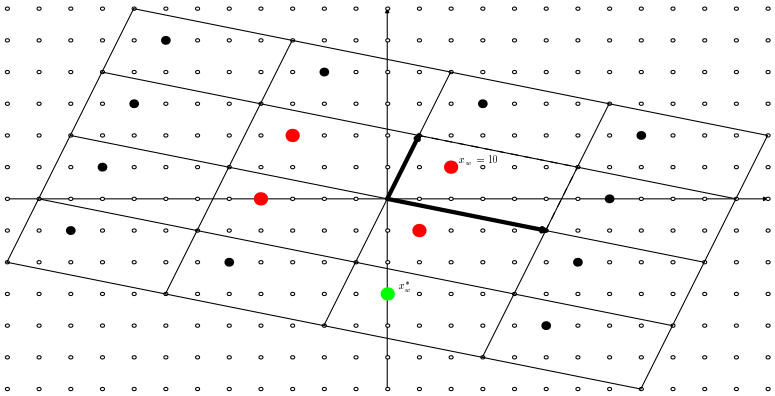
C: Celoštevilska mreža $\mathbb{Z}_2^2$. Za dani vrednosti $u, v \in V_n$ naj bo $w = v - u \pmod{n}$ in naj bo $i \in \{1, -1, 2, -2\}$. Poišči po ℓ_1 normi najmanjši element množice $X_w \cap \mathbb{Z}_{-i}^2$.

Ker so zgornji opisi problema iskanja omejene najkrajše poti podobni opisom iskanja najkrajše poti v neusmerjenem krožnem grafu, veliko dodatne razlage ne potrebujejo. Pomen pogoja $\text{sign}(i) * x_{|i|} \leq 0$ v **B** smo opisali v razdelku 3.12, kjer je tudi razlaga, zakaj v **C** rešitve iščemo le med elementi množice $X_w \cap \mathbb{Z}_{-i}^2$. Brez dodatne razlage lahko torej ugotovimo, da vse tri zgornje formulacije opisujejo isti problem. Omenimo še, da so ob predpostavki $\gcd(n, h_1, h_2) = 1$ zgornji problemi vedno rešljivi, včasih obstaja celo več rešitev.

> Primer 3.13-I. V grafu $G^{-2}(6;\ \pm 1, \pm 2)$ sprehoda $(-2, 0)$ (dve kratki povezavi nazaj) in $(0, 2)$ (dve dolgi povezavi naprej) povezujeta točki 0 in 4. Ker noben od njiju ne uporablja dolge povezave nazaj in ker krajšega sprehoda med točkama 0 in 4 v tem grafu ni, sta oba sprehoda rešitev problema iskanja omejene najkrajše poti med 0 in 4 v grafu $G^{-2}(6;\ \pm 1, \pm 2)$.

3.13.1 Na poti do rešitve

Podobno kot v usmerjenem in neusmerjenem primeru bomo tudi v omejenem primeru reševali problem **C**, to je problem iskanja po ℓ_1 normi najmanjšega elementa v delu celoštevilske mreže. Pri reševanju bomo uporabili ugotovitve, ki smo jih pridobili pri reševanju neomejenega problema. Najprej bomo predstavili dva ponesrečena poskusa reševanja – intuitivno smo namreč med raziskavo nekoliko prehitro sklepali in postavili najprej eno in nato (ko se je ta izkazala za neresnično) še drugo hipotezo. Ko smo našli pomanjkljivost tudi v drugi hipotezi, smo postavili tretjo. Ta hipoteza, ki se je končno izkazala za pravilno in jo zato v tem razdelku imenujemo kar trditev, podaja navodilo, kako poiskati omejene najkrajše poti. Predstavili jo bomo, skupaj z dokazom, takoj za predstavitvijo neuspelih poskusov.



Slika 3.13-I: Protiprimer za hipotezo 1: v grafu $G^1(11;\ \pm 3, \pm 4)$ s pakirano bazo $\{a, b\} = \{(5, -1), (1, 2)\}$ najkrajša pot (0,-3) med 0 in $w = 10$ ni med baznimi elementi leve polravnine

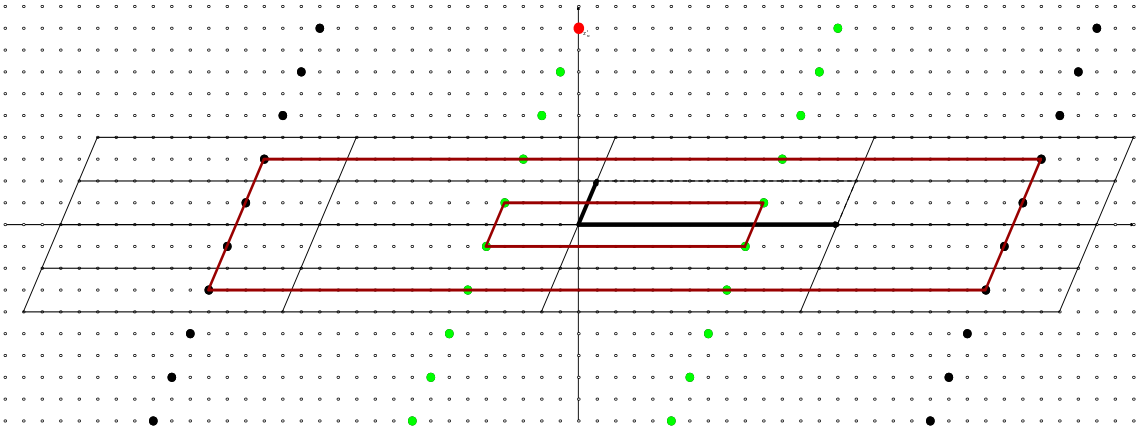
Prvi poskus – optimizem
Problem (neomejenih) najkrajših poti smo rešili s pomočjo pakiranih baz in sicer takole: (a) poiskali smo pakirano bazo (a, b), (b) poiskali smo $x_w \in [a, b)$ – predstavnika množice X_w v baznem paralelogramu, (c) med elementi množice $C_1^w = \{x_w, x_w - a, x_w - b, x_w - (a + b)\}$ smo izbrali tistega z najmanjšo ℓ_1 normo. Pri prvem poskusu iskanja omejene najkrajše poti smo želeli uporabiti kar iste (ali pa nekoliko prirejene) izreke in rešitev iskati med elementi množice $C_1^w \cap \mathbb{Z}^2_{-i}$.

Hipoteza 1. Naj bo $\{a, b\}$ pakirana baza, w nek element množice V_n in x_w osnovni bazni element množice X_w. Omejeno najkrajšo pot med 0 in w v grafu $G^i(n;\ \pm h_1, \pm h_2)$ iščemo med štirimi baznimi elementi, t.j., omejeno najkrajšo pot predstavlja po ℓ_1 normi najmanjši element množice $\{x_w, x_w - a, x_w - b, x_w - (a + b)\} \cap \mathbb{Z}^2_{-i}$.

Hipoteza 1 je neresnična. Sliko 3.13-I prikazuje enega od mnogih protiprimerov: v grafu $G^1(11;\ \pm 3, \pm 4)$ s pakirano bazo $\{a, b\} = \{(5, -1), (1, 2)\}$ najkrajša pot (0,-3) med 0 in $w = 10$ ni med baznimi elementi leve polravnine (bazni elementi so na sliki prikazani z debelimi črnimi pikami, pot (0,-3) pa z debelo sivo piko).

Drugi poskus – naivnost
Nepravilnost hipoteze 1 lahko pripišemo temu, da pregleda premalo kandidatov, ali pa se odloči za napačne. Z drugo hipotezo bomo pregledali večjo množico in sicer poleg že omenjenih štirih baznih elementov (rešitev se kljub vsemu največkrat skriva med njimi) bomo pregledali še njihove najbližje sosede. Kot smo videli v razdelku 3.7, so bazni elementi prvi, njihovi najbližji sosedje pa drugi cikel množice X_w.

Hipoteza 2. Naj bo $\{a, b\}$ pakirana baza, $w \in V_n$ in x_w osnovni bazni element množice X_w. Omejeno najkrajšo pot med točkama 0 in w grafa $G^i(n;\ \pm h_1, \pm h_2)$



Slika 3.13-II: Protiprimer za hipotezo 2: v grafu $G^{-1}(28;\ \pm 2, \pm 13)$ s pakirano bazo $\{a, b\} = \{(14, 0), (1, 2)\}$ najkrajša pot (0,9) med 0 in $w = 5$ leži šele v petem ciklu

najdemo v preseku prvega in drugega cikla množice X_w s polravnino $\mathbb{Z}^2_{-i}$, oziroma, $p_i^{(u,v)} \in (C_1^w \cup C_2^w) \cap \mathbb{Z}^2_{-i}$.

Za primer, ki smo ga na sliki 3.13-I prikazali kot protiprimer hipoteze 1, kot tudi za mnoge druge primere, hipoteza 2 drži, na splošno pa ne. Protiprimer zanjo prikazuje slika 3.13-II, na kateri je prikazana mreža za graf $G^{-1}(28;\ \pm 2, \pm 13)$, pri katerem najkrajša pot med točkama 0 in 5 ne leži niti v prvem niti v drugem ciklu, pač pa šele v petem.

Tretji poskus – zmaga!
Hipoteza 2 sama po sebi ni pomembna, saj ni le neresnična, je tudi zavajajoča. Kot smo namreč videli v protiprimeru, lahko rešitev problema leži v precej oddaljenem ciklu in zato selektivno pregledovanje ciklov, kot svetuje hipoteza, ni perspektivno opravilo. Kljub temu pa je prav ta hipoteza pripomogla k temu, da smo našli pravilno rešitev problema. Iz njenih protiprimerov (tudi iz tistega na sliki 3.13-II) se vidi, zakaj ni resnična, oziroma, kaj bi morala trditi, da bi bila. Če pogledamo sliko 3.13-II, vidimo, da je rešitev problema prvi element traku $t(x_w - (a+b), b)$ v desni polravnini $\mathbb{Z}^2_1$. Kot se bo izkazalo, tudi v splošnem primeru velja nekaj podobnega, in sicer, da rešitev problema omejenih najkrajših poti vedno leži v preseku baznih trakov in odgovarjajoče polravnine. Formalno to ugotovitev podaja naslednja trditev.

Trditev 3.13-I. *Naj bodo $\{a, b\}$ pakirana baza modula X za graf $G(n;\ \pm h_1, \pm h_2)$, $i \in I$, $w \in V_n$ in x_w osnovni bazni element. Nadalje naj T_w^i predstavlja množico točk, ki ležijo v preseku baznih trakov $t(x_w, a), t(x_w, b), t(x_w - a, b)$ in $t(x_w - b, a)$ s polravnino $\mathbb{Z}^2_i$. Tedaj najmanjši element množice X_w^i leži v T_w^i.*

3.13.2 Rešitev problema

Preden se lotimo formalnega dokaza pravilnosti trditve 3.13-I, si oglejmo, kakšen je pomen te trditve v praksi, oziroma, kako lahko z njeno pomočjo hitro poiščemo omejene najkrajše poti. Trditev 3.13-I pravi, da lahko pri reševanju tega problema pregledamo le elemente baznih trakov v odgovarjajoči polravnini. Čeprav je teh elementov neskončno mnogo, je naloga "pregledovanja" preprosta. Kot smo videli v razdelku 3.9.1 o minimalni projekciji, se da na vsakem traku v konstantnem času poiskati točko, ki ima v dani polravnini najmanjšo normo. Ker vemo, da je množica kandidatov T_w^i sestavljena iz največ štirih poltrakov (po definiciji je T_w^i unija štirih baznih poltrakov, v posebnih primerih pa se lahko zgodi, da je kateri od njih prazen), je postopek za iskanje omejene najkrajše poti takle: poišči najmanjše elemente poltrakov $t^i(x_w, a)$, $t^i(x_w, b)$, $t^i(x_w - a, b)$ ter $t^i(x_w - b, a)$ in med njimi izberi najmanjšega. Najmanjši elementi na omenjenih poltrakovih so minimalne projekcije elementov x_w in $x_w - (a + b)$ v smeri baznih vektorjev, zato se pravkar opisani postopek lahko zapiše tudi v obliki, ki je podana v algoritmu 3.13-I.

Algoritem 3.13-I	$2RSPMP(n, h_1, h_2, w, i)$

Računanje omejene najkrajše poti med 0 in w v $G^i(n;\ \pm h_1, \pm h_2)$

$2RSPMP(n, h_1, h_2, w, i)$:

1: $\{a, b\}$:= pakirana baza modula X v $G(n;\ \pm h_1, \pm h_2)$;

2: x_w := osnovni bazni element;

3: $P := \{P^{-i}(x_w, a), P^{-i}(x_w, b), P^{-i}(x_w - a, b), P^{-i}(x_w - b, a)\}$;

4: **return** $\min_{\ell_1} P$.

Algoritma 3.11-III za računanje najkrajših poti in 3.13-I za računanje omejenih najkrajših poti sta si zelo podobna. Oba namreč končno rešitev iščeta med štirimi kandidati – prvi med baznimi elementi, drugi pa med projekcijami teh elementov v smereh a in b v odgovarjajočo polravnino. Res, saj za vsak $i \in I$ velja

$$\begin{aligned} P^i(x_w, a) &= P^i(x_w - a, a) \\ P^i(x_w - (a + b), a) &= P^i(x_w - b, a) \\ P^i(x_w, b) &= P^i(x_w - b, b) \\ P^i(x_w - (a + b), b) &= P^i(x_w - a, b) \end{aligned}$$

in so zato projekcije baznih elementov v smereh baznih vektorjev enake elementom tretje točke algoritma 3.13-I, med katerimi iščemo rešitev problema. Tudi časovna zahtevnost obeh algoritmov je enaka, saj je računanje minimalnih projekcij preprosta operacija, ki traja le konstantno mnogo časa.

Kongruenčna enačba z dvema dodatnim pogojema

V uvodu poglavja smo videli, da je iskanje omejene najkrajše poti med točkama 0 in w dvokrožnega grafa $G(n;\ \pm h_1, \pm h_2)$ ekvivalentno iskanju take rešitve

$$x_1 h_1 + x_2 h_2 \equiv w \pmod{n}, \tag{3.11}$$

ki med vsemi pari $(x_1, x_2) \in \mathbb{Z}_2^2$, za katere velja $\text{sign}(i) * x_{|i|} \leq 0$, minimizira $|x_1| + |x_2|$. S predstavitvijo algoritma 3.13-I za izračun omejene najkrajše poti smo torej hkrati predstavili tudi algoritem za reševanje kongruenčne enačbe 3.11 ob omenjenih dodatnih pogojih.

3.13.3 Strategija dokazovanja

Trditev 3.13-I, ki pravi, da najmanjši element množice X_w^i leži v T_w^i, bomo dokazali tako, da bomo dokazali naslednjo (močnejšo) trditev:

$$\text{za } \forall x \in X_w^i \text{ obstaja tak } y \in T_w^i, \text{ da velja } ||y|| \leq ||x||. \tag{3.12}$$

Za elemente x, ki že ležijo na enem od trakov, ta trditev seveda drži (iskani y je kar x), zato se bomo posvetili predvsem elementom X_w^i, ki ne ležijo v T_w^i, torej elementom množice

$$D_w^i := X_w^i - T_w^i.$$

Pokazali bomo, da obstaja taka transformacija nad elementi množice D_w^i, ki ima lastnosti:

(a) slika elementov iz D_w^i leži v X_w^i,

(b) elementi množice D_w^i se s to transformacijo "približajo" množici T_w^i in

(c) norma se pri transformaciji ne poveča.

Kaj točno pomeni "približati" množici trakov (točka b), bomo videli kasneje, za zdaj si to lahko predstavljamo takole: iskana transformacija ima lastnost, da zmanjša neko pozitivno celoštevilsko vrednost, ki pripada elementom domene, za najmanj ena. Več zaporednih transformacij postopno zmanjša to vrednost na 0, to je na vrednost, ki pripada elementom množice T_w. Z drugimi besedami: več zaporednih transformacij poljuben element domene preslika v množico T_w^i, poleg tega pa norma končnega elementa ni večja od norme začetnega elementa, saj se na posameznem koraku ta ne poveča (točka c). Obstoj takšne transformacije torej dokaže pravilnost trditve, zapisane v (3.12), in posledično tudi trditve 3.13-I.

Približevanje baznim trakovom

Ustavimo se še pri pogoju (b) iskane transformacije in povejmo, kaj natančno pomeni izraz *približati baznim trakovom*. V ta namen najprej definirajmo pojem *razdalja točke od baznih trakov* – približati baznim trakovom bo potem pomenilo zmanjšati to razdaljo za najmanj ena. Rekli bomo, da je $x = x_w + m_x a + n_x b \in X_w$ za $r_n(x)$ oddaljen od a-baznih trakov (torej od trakov a_0 in a_{-1}), če velja

$$x + r_n(x) * b \in T_w \quad \text{ali} \quad x - r_n(x) * b \in T_w$$

in če za vsak $t \in \mathbb{N},\ t \neq r_n(x)$, za katerega je $x + t * b \in T_w$ ali $x - t * b \in T_w$, velja $t > r_n(x)$. Z drugimi besedami: oddaljenost od a-baznih trakov je definirana kot število potrebnih korakov v smeri b ali $-b$, da od elementa x pridemo do nekega elementa na traku a_0 ali a_{-1}. Elementom prvega in drugega prekata je bližji trak a_0, njihova oddaljenost od tega traku pa je enaka n_x, elementom tretjega in četrtega prekata pa je bližji trak a_{-1}, njihova oddaljenost od njega pa je $|n_x + 1|$. Ker za prve velja $n \geq 0$, za druge pa $n \leq -1$, velja

$$r_n(x) = \min\{|n_x|, |n_x + 1|\}.$$

Na enak način definiramo tudi oddaljenost elementa x od b-baznih trakov. Če to razdaljo označimo z $r_m(x)$, lahko napišemo $r_m(x) = \min\{|m_x|, |m_x + 1|\}$ (pravilnost formule dokažemo s podobnimi argumenti, kot smo to storili pri $r_n(x)$). Razdaljo točke $x \in X_w$ od baznih trakov T_w bomo sedaj definirali kot produkt razdalij te točke od a-baznih in b-baznih trakov.

Definicija 3.13-II. *Za vsak* $x \in X_w$ razdaljo od baznih trakov $r(x)$ *definiramo kot*

$$r(x) := r_n(x)\ r_m(x).$$

Razdalja točke $x \in X_w$ od baznih trakov se zmanjša (t.j. točka se približa baznim trakovom) brž, ko se zmanjša ena od vrednosti r_n ali r_m, druga pa se pri tem ne poveča. Za vsak $x \in T_w$ je razdalja $r(x)$ enaka 0.

Univerzalna transformacija
Iskanje *ne-sestavljene* univerzalne transformacije, ki bi bila dobra v vseh primerih, je nesmiselno početje, saj taka transformacija ne obstaja. Na primer: elemente prvega prekata transformacija $x \mapsto x - a - b$ približa baznim trakovom, elemente četrtega prekata pa ta transformacija od baznih trakov oddalji. Izkaže se, da na "uspešnost" transformacije (to je, ali transformacija res približa trakovom in pri tem ne poveča norme) poleg položaja elementa, na katerega ta deluje, vpliva še število i, tip baze in, v nekaterih primerih, kot med baznimi vektorji in abscisno oz. ordinatno osjo. Pri dokazovanju trditve 3.13-I bomo torej ločili več primerov in za vsakega izmed njih poiskali primerno transformacijo.

Zapis ekvivalentne trditve
Trditev 3.13-I lahko zapišemo v obliki, ki je za dokazovanje primernejša. Glede na vse povedano bo dokaz te trditve tudi dokaz trditve 3.13-I.

Trditev 3.13-III. *Naj bo* $\{a, b\}$ *pakirana baza modula* X *za graf* $G(n;\ \pm h_1, \pm h_2)$, $w \in V_n$ *in* x_w *osnovni bazni element. Za vsako polravnino* $i \in I$*, za vsak prekat* $P \in \{1, 2, 3, 4\}$ *in za vsak element* $x = x_w + m_x a + n_x b \in (P)^i_w$ *obstaja transformacija* $\tau : X_w \to X_w$*, za katero velja:*

a. slika elementov $(P)^i_w$ *je v* $[P]^i_w$*,*

b. $r(\tau(x)) < r(x)$ *in*

c. $||\tau(x)|| \leq ||x||$.

3.13.4 Dokaz pravilnosti

Trditev 3.13-III bomo najprej dokazali za $i = 1$, nato bomo pokazali, kako lahko ta dokaz prilagodimo tudi za $i \in \{-1, 2, -2\}$. Pri $i = 1$ se bomo najprej posvetili prvemu prekatu ($P = 1$) in pokazali, da za vsak x vsaj ena od transformacij $x \mapsto x - a$ in $x \mapsto x - b$ izpolnjuje vse tri pogoje. Za $P = 2$ bomo nato za posamezen tip pakirane baze (11 podprimerov) poiskali primerno transformacijo. Za primer $P = 3$ bomo pokazali, da je simetričen primeru $P = 2$, za $P = 4$ pa, da je množica $(4)^1_w$ prazna in zato trditev gotovo drži. S tem bosta trditvi 3.13-I in 3.13-III dokazani. Da bi bralcu olajšali branje tega obsežnega dokaza, v tabeli 3.13-I podajamo razporeditev posameznih delov dokaza (glede na vrednost i in P ter glede na tip pakirane baze) po straneh.

i	P	tip pakirane baze	stran
1	1	A ... K	82
1	2	A, B, C	83
1	2	G, I	83
1	2	D, E	84
1	2	F, H	86
1	2	J, K	87
1	3	A ... K	90
1	4	A ... K	91
-1	1, -1, 2, -2	A ... K	92
2, -2	1, -1, 2, -2	A ... K	92

Tabela 3.13-I: Razporeditev posameznih delov dokaza trditve 3.13-III po straneh

Pri dokazovanju pravilnosti trditve 3.13-III bomo potrebovali naslednjo lemo.

Lema 3.13-IV. *Naj bosta* $\{a, b\}$ *pakirana baza modula* X *in* w *ena od oznak mreže* $\mathbb{Z}_2^2$ *za graf* $G(n;\ \pm h_1, \pm h_2)$. *Potem za vsak* $x \in (1)^1_w$ *velja*

$$||x|| \geq \min(||x - a||, ||x - b||).$$

Dokaz. Lemo 3.13-IV bomo dokazali s protislovjem tako, da bomo pokazali, da v primeru, ko hkrati velja $||x|| < ||x-a||$ in $||x|| < ||x-b||$, baza $\{a, b\}$ ni pakirana. Oglejmo si najprej vrednosti norm $||x||$, $||x - a||$ in $||x - b||$. Vsi trije elementi ležijo v $\mathbb{Z}_1^2$, zato je njihova prva koordinata nenegativna in lahko v zapisu izpustimo absolutno vrednost ter dobimo

⎤82
⋮

$$\begin{aligned} ||x|| &= x_{w1} + ma_1 + nb_1 + |x_{w2} + ma_2 + nb_2|, \\ ||x-a|| &= x_{w1} + (m-1)a_1 + nb_1 + |x_{w2} + (m-1)a_2 + nb_2|, \\ ||x-b|| &= x_{w1} + ma_1 + (n-1)b_1 + |x_{w2} + ma_2 + (n-1)b_2|. \end{aligned}$$

Vrednosti razlik $||x|| - ||x-a||$ in $||x|| - ||x-b||$ so potem

$$||x|| - ||x-b|| = b_1 + |\underbrace{x_{w2} + ma_2 + nb_2}_{\alpha}| - |\underbrace{x_{w2} + ma_2 + (n-1)b_2}_{\beta}|, \quad (3.13)$$

$$||x|| - ||x-a|| = a_1 + |\underbrace{x_{w2} + ma_2 + nb_2}_{\alpha}| - |\underbrace{x_{w2} + (m-1)a_2 + nb_2}_{\gamma}|. \quad (3.14)$$

Vrednosti razlik (3.13) in (3.14) so odvisne od predznakov členov α, β in γ, ti pa so odvisni od lege elementov x, $x-a$ in $x-b$ glede na abscisno os (pod ali nad njo). Glede na te lege bomo ločili 8 primerov in dokaz nadaljevali za vsakega posebej. Primere bomo označili takole:

A: x, $x-a$ in $x-b$ so nad abscisno ..... $\alpha \geq 0,\ \beta \geq 0,\ \gamma \geq 0,$

A': x, $x-a$ in $x-b$ so pod abscisno ..... $\alpha \leq 0,\ \beta \leq 0,\ \gamma \leq 0,$

B: x je nad, $x-a$ in $x-b$ pa pod abscisno ..... $\alpha \geq 0,\ \beta \leq 0,\ \gamma \leq 0,$

B': x je pod, $x-a$ in $x-b$ pa nad abscisno ..... $\alpha \leq 0,\ \beta \geq 0,\ \gamma \geq 0,$

C: x in $x-b$ sta pod, $x-a$ pa nad abscisno ..... $\alpha \leq 0,\ \beta \geq 0,\ \gamma \leq 0,$

C': x in $x-a$ sta pod, $x-b$ pa nad abscisno ..... $\alpha \leq 0,\ \beta \leq 0,\ \gamma \geq 0,$

D: x in $x-a$ sta nad, $x-b$ pa pod abscisno ..... $\alpha \geq 0,\ \beta \geq 0,\ \gamma \leq 0,$

D': x in $x-b$ sta nad, $x-a$ pa pod abscisno ..... $\alpha \geq 0,\ \beta \leq 0,\ \gamma \geq 0.$

Vrednosti razlik (3.13) in (3.14) za posamezen primer prikazuje tabela 3.13-II.

Primer A ($\alpha \geq 0,\ \beta \geq 0,\ \gamma \geq 0$)**:** recimo, da velja $||x|| - ||x-a|| < 0$ in $||x|| - ||x-b|| < 0$, oziroma (glej tabelo 3.13-II), $a_1 + a_2 < 0$ in $b_1 + b_2 < 0$. Ker sta a_1 in b_1 nenegativna, velja

$$a_2 < 0 \text{ in } b_2 < 0 \quad (3.15)$$

ter

$$|a_2| > a_1 \text{ in } |b_2| > b_1. \quad (3.16)$$

primer	α	β	γ	$\|\|x\|\| - \|\|x-a\|\|$	$\|\|x\|\| - \|\|x-b\|\|$
A	$+$	$+$	$+$	$a_1 + a_2$	$b_1 + b_2$
A'	$-$	$-$	$-$	$a_1 - a_2$	$b_1 - b_2$
B	$+$	$-$	$-$	$a_1 - a_2 + 2x_2$	$b_1 - b_2 + 2x_2$
B'	$-$	$+$	$+$	$a_1 + a_2 - 2x_2$	$b_1 + b_2 - 2x_2$
C	$-$	$+$	$-$	$a_1 + a_2 - 2x_2$	$b_1 - b_2$
C'	$-$	$-$	$+$	$a_1 - a_2$	$b_1 + b_2 - 2x_2$
D	$+$	$+$	$-$	$a_1 + a_2$	$b_1 - b_2 + 2x_2$
D'	$+$	$-$	$+$	$a_1 - a_2 + 2x_2$	$b_1 + b_2$

Tabela 3.13-II: Vrednosti razlik (3.13) in (3.14) glede na lego elementov x, $x-a$ in $x-b$ v $\mathbb{Z}_2^2$ (nad ali pod abscisno osjo)

To pa pomeni, da vektorja a in b ležita v 4. kvadrantu (enačba 3.15), kot med njima in ordinatno osjo pa je manjši od 45° (enačba 3.16), torej je kot med vektorjema a in b manjši od 45°, kar pa je po trditvi 3.10-XI v nasprotju s predpostavko o pakiranosti baze. $\multimap\!\!*\!\!-$

Primer A′ ($\alpha \le \mathbf{0},\ \beta \le \mathbf{0},\ \gamma \le \mathbf{0}$): dokaz za ta primer poteka podobno (simetrično) kot dokaz za primer A. Iz $a_1 - a_2 < 0$ in $b_1 - b_2 < 0$ (glej tabelo 3.13-II) izpeljemo, da a in b ležita v prvem kvadrantu in da je kot med njima manjši od 45°. $\multimap\!\!*\!\!-$

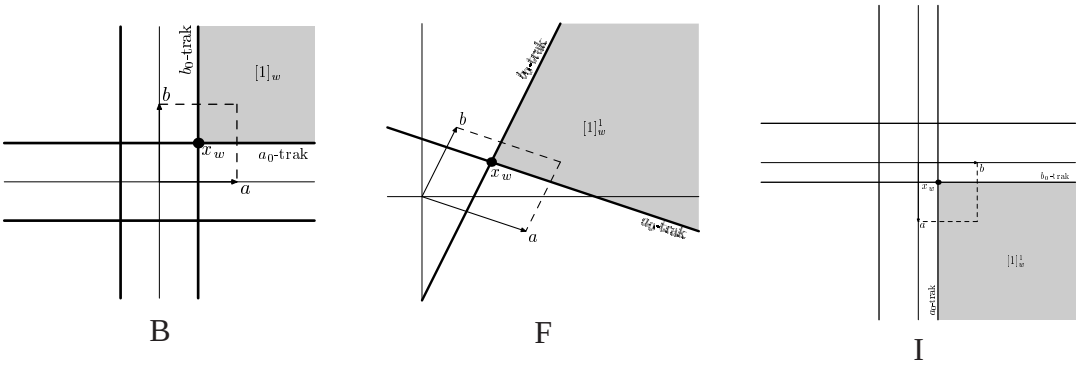
Primer B ($\alpha \ge \mathbf{0},\ \beta \le \mathbf{0},\ \gamma \le \mathbf{0}$): iz predpostavke $||x|| - ||x-a|| = a_1 - a_2 + 2x_2 < 0$ sledi $a_2 > 0$, saj sta tako a_1 kot x_2 nenegativni števili. Nadalje vidimo, da $a_2 > a_1 + 2x_2$ oziroma $a_2 > a_1$. Podobno iz predpostavke $||x|| - ||x-b|| = b_1 - b_2 + 2x_2 < 0$ izpeljemo $b_2 > 0$ in $b_2 > b_1$. Za bazo s temi lastnostmi pa smo v primeru A' videli, da ni pakirana. $\multimap\!\!*\!\!-$

Primer B′ ($\alpha \le \mathbf{0},\ \beta \ge \mathbf{0},\ \gamma \ge \mathbf{0}$): podobno kot pri B iz $a_1 + a_2 - 2x_2 < 0$ izpeljemo $a_2 < 0$ in $|a_2| > a_1$, iz $b_1 + b_2 - 2x_2 < 0$ pa $b_2 < 0$ in $|b_2| > a_1$. Baza s takimi lastnostmi pa ni pakirana (glej primer A). $\multimap\!\!*\!\!-$

Primer C ($\alpha \le \mathbf{0},\ \beta \ge \mathbf{0},\ \gamma \le \mathbf{0}$): iz neenakosti $a_1 + a_2 - 2x_2 < 0$ sledi $a_2 < 0$ in $|a_2| > a_1$, iz $b_1 - b_2 < 0$ pa $b_2 > 0$ in $b_2 > b_1$. Baza ima v tem primeru torej naslednje lastnosti: vektor a leži v četrtem, vektor b pa v prvem kvadrantu, poleg tega sta tako kot med a in absciso kot tudi kot med b in absciso večja od 45°. Taka baza pa po lemi 3.10-IX ni pakirana. $\multimap\!\!*\!\!-$

Primer C′ ($\alpha \le \mathbf{0},\ \beta \le \mathbf{0},\ \gamma \ge \mathbf{0}$): dokaz poteka enako kot pri C. Iz neenakosti $a_1 - a_2 < 0$ in $b_1 + b_2 - 2x_2 < 0$ dobimo iste pogoje za bazo kot pri C, le da sta vlogi vektorjev a in b zamenjani. $\multimap\!\!*\!\!-$

Primer D ($\alpha \ge \mathbf{0},\ \beta \ge \mathbf{0},\ \gamma \le \mathbf{0}$) **in D′** ($\alpha \ge \mathbf{0},\ \beta \le \mathbf{0},\ \gamma \ge \mathbf{0}$): podobno kot pri C in C' dobimo tudi pri D in D' bazo, katere en vektor je v prvem, en pa v četrtem kvadrantu,



B F I

Slika 3.13-III: Oblika in lega območja $[1]_w$ pri pakiranih bazah tipa B, F in I

79 kot med njima pa je večji od 90°. —×— □

Dokaz trditve 3.13-III za primer $P = 1$ ↓

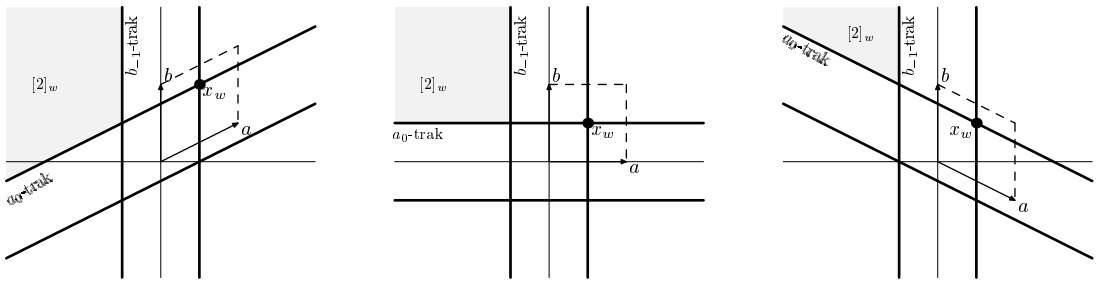
83 *Dokaz.* (trditev 3.13-III, $i = 1,\ P = 1$; glej sliko 3.13-III)
Iskanje primerne transformacije za primer $i = 1,\ P = 1$ precej olajša dejstvo, da tako celotna domena kot celotna kodomena iskane transformacije ležita, kot je razvidno iz tipičnih primerov na sliki 3.13-III, v desni polravnini $\mathbb{Z}_1^2$. O tem se zlahka tudi formalno prepričamo, saj veljajo naslednje ugotovitve:

1. ker sta vektorja a in b v prvem ali četrtem kvadrantu (torej v $\mathbb{Z}_1^2$), je v $\mathbb{Z}_1^2$ tudi osnovni bazni element x_w, torej je prva koordinata elementa x_w nenegativna ($x_{w1} \geq 0$),

2. ker sta a in b v $\mathbb{Z}_1^2$, sta njuni prvi koordinati nenegativni, torej je $a_1 \geq 0$ in $b_1 \geq 0$,

3. vsak $x = (x_1, x_2) \in [1]_w$ lahko pišemo v obliki $x = x_w + m_x a + n_x b$, pri čemer velja $m_x \geq 0$ in $n_x \geq 0$ (glej karakteristike prekatov, tabela 3.7-I na strani 45); za prvo koordinato elementa x zato velja $x_1 = x_{w1} + m_x a_1 + n_x b_1 \geq 0$, torej je $x \in \mathbb{Z}_1^2$.

Poiščimo zdaj transformacijo, ki bo izpolnjevala pogoje (a), (b) in (c) trditve 3.13-III. Če naj transformacija nad elementi prvega prekata izpoljnjuje pogoj (b), mora zmanjšati eno od količin m_x ali n_x (in druge ne povečati), poleg tega pa, če želimo, da je izpolnjen še pogoj (a), to zmanjšanje ne sme biti preveliko, to je, m_x in n_x se ne smeta zmanjšati za več kot ena. Edine transformacije torej, ki za vsak $x \in (1)_w^1$ izpolnjujejo pogoja (a) in (b), so $t_1 : x \mapsto x - a$, $t_2 : x \mapsto x - b$ in $t_3 : x \mapsto x - (a + b)$. Kako pa je s pogojem (c) za te transformacije? Poglejmo, na primer, transformacijo t_2. Če so b, x in $x - b$ vsi elementi prvega kvadranta, potem velja $||x-b|| < ||x||$; če pa sta b in x v prvem, $x-b$ pa v četrtem kvadrantu in če je poleg tega kot med b in absciso večji od 45°, je lahko $||x - b|| > ||x||$. Transformacija t_2 torej v določenih primerih izpolnjuje pogoj (c), ne pa v vseh; podobno se izkaže tudi za t_1 in t_3. Nobena od treh možnih kandidatk torej ne izpolnjuje pogoja (c) v vseh primerih, izkaže pa se, da se t_1 in t_2 dopolnjujeta, to je, da v vsakem primeru



A B C

Slika 3.13-IV: Oblika in lega območja $[2]_w$ pri pakiranih bazah tipa A, B in C

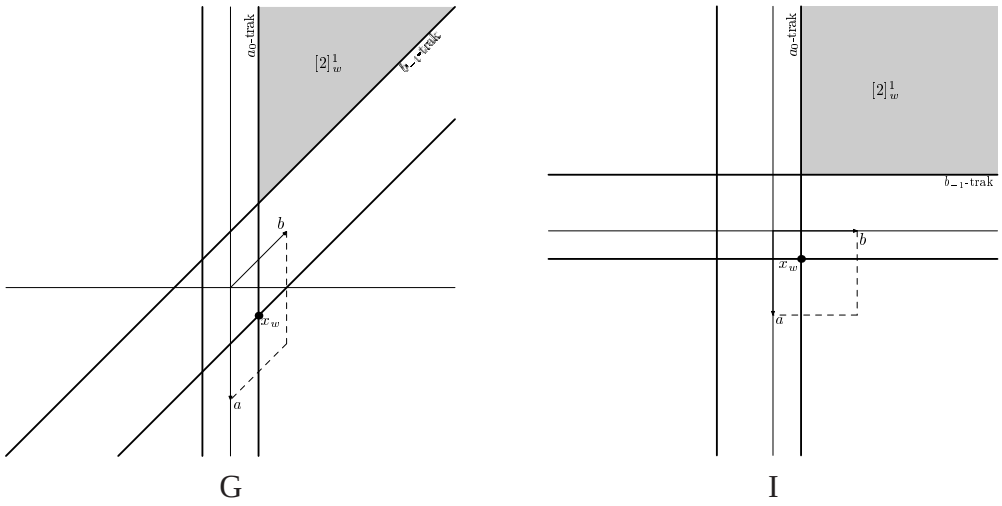
vsaj ena od njiju izpolnjuje usodni pogoj (c). Res, saj po lemi 3.13-IV za vsak $x \in (1)^1_w$ velja $||x|| \geq \min(||x-a||, ||x-b||)$. Za vsak $x \in (1)^1_w$ torej ena od transformacij t_1 in t_2 izpolnjuje vse tri pogoje. □ ⌋82

Dokaz trditve 3.13-III za primer $P = 2$ ↓

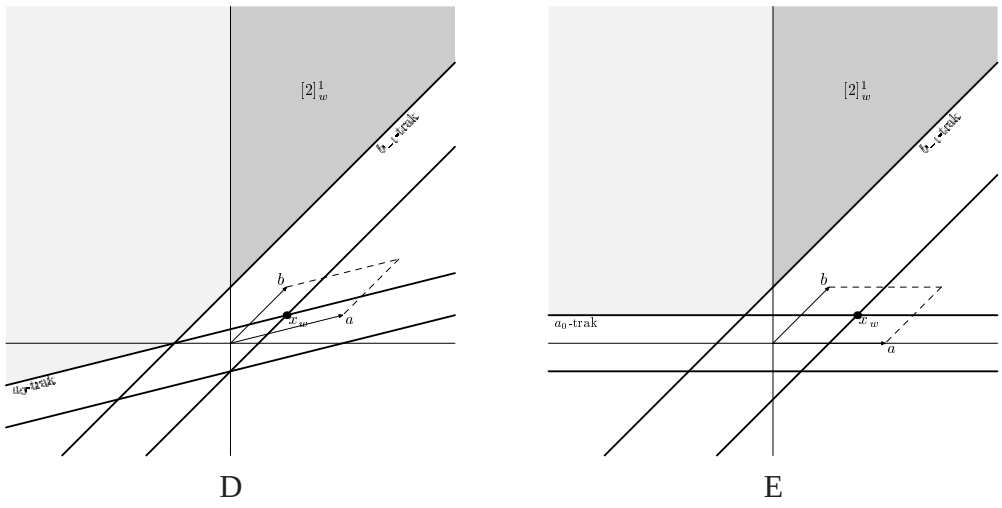
Bistvena razlika med primeroma $P = 1$ in $P = 2$ je v legi območja, ki ga v posameznem primeru obravnavamo. V prvem primeru $[1]_w$ v celoti leži v $\mathbb{Z}^2_1$, v drugem primeru pa temu ni tako - območje $[2]_w$ lahko, odvisno od tipa pakirane baze, seka prvi, drugi, tretji in četrti kvadrant. V prvem primeru smo iskali transformacije, ki so približale baznim trakovom, v drugem primeru pa bodo morale transformacije poleg tega zagotoviti še, da slika elementov ostane v $\mathbb{Z}^2_i$. Zaradi te dodatne zahteve je dokaz za primer $P = 2$ bolj zapleten, kot je bil v prejšnjem primeru. Ker lega območja $[2]_w$ igra bistveno vlogo pri iskanju primerne transformacije in ker je ta odvisna od tipa pakirane baze, primer $P = 2$ dokažemo za vsak tip baze posebej. Izkaže se, da so od tako dobljenih enajstih primerov trije trivialni (za baze tipa A, B in C), dva dokaza sta precej preprosta (G in I), preostalih šest primerov pa dokažemo po dva in dva skupaj (D in E, F in H ter J in K). Preden se lotimo dokazov za posamezen primer, si oglejmo še nekaj splošnih lastnosti. Območje $[2]_w$ omejujeta trakova a_0 in b_{-1}, elementi tega območja pa so oblike $x = x_w + m_x a + n_x b$ pri $m_x < 0$ in $n_x \geq 0$. Približevalne transformacije za primer $P = 2$, to je transformacije, ki izpolnjujejo pogoj (b) trditve 3.13-III, so oblike $x \mapsto x + ta - sb$ pri $t, s \geq 0$. Ponovno se dogovorimo še to, da vektorja pakirane baze označimo tako, da a leži pod b.

Dokaz. (trditev 3.13-III, $i = 1,\ P = 2$,baze tipa A, B in C; glej sliko 3.13-IV)
Pri bazah tipa A, B in C vektor b leži na ordinatni osi v prvem kvadrantu. Elementi območja $[2]_w$ torej ležijo v desni polravnini $\mathbb{Z}^2_{-1}$. Res, saj za vsak $x \in [2]_w$ velja $x_1 = x_{w1} + m_x a_1$ pri $m_x < 0$. Ker je $a_1 < x_{w1}$, je $x_1 < 0$. Območje $(2)^1_w$ je torej prazno, zato so vse zahteve trditve na prazno izpolnjene. □

Dokaz. (trditev 3.13-III, $i = 1,\ P = 2$, baze tipa G in I; glej sliko 3.13-V)
Za primera G in I bomo pokazali, da transformacija $\tau : x \mapsto x - b$ izpolnjuje vse tri pogoje izreka. Najprej vidimo, da pri bazah tipa G in I celotno območje $[2]_w$ leži v prvem



Slika 3.13-V: Oblika in lega območja $[2]_w$ pri pakiranih bazah tipa G in I



Slika 3.13-VI: Oblika in lega območja $[2]_w$ pri pakiranih bazah tipa D in E

kvadrantu. Res, vse točke tega območja ležijo desno in nad točko $x_w - a$, ta pa leži v prvem kvadrantu. Iz tega sledi, da transformacija τ izpolnjuje pogoj (a), saj slika elementov iz $(2)^1_w$ gotovo leži v $[2]_w$ (po definiciji transformacije τ), torej v $[2]^1_w$, saj smo ravnokar videli, da je $[2]^1_w = [2]_w$. Da τ izpolnjuje pogoj (b), je očitno, saj elementom x iz $(2)_w$ zmanjša vrednost števila n_x za ena, posledično pa s tem za ena zmanjša tudi vrednost razdalje r_n; vrednost razdalje r_m se ne spremeni. Prav tako je očitno tudi izpolnjevanje pogoja (c), saj je $||\tau(x)|| = ||x|| - ||b||$, oziroma $||\tau(x)|| < ||x||$. □

Dokaz. (trditev 3.13-III, $i = 1,\ P = 2$, baze tipa D in E; glej sliko 3.13-VI)
Dokažimo najprej, da celotno območje $[2]^1_w$ leži v zgornji polravnini. Območje $[2]_w$ omejujeta a_0 in b_{-1}-trak. Označimo s S sečišče b_{-1} traku z ordinatno osjo. Ker b_{-1}-trak leži

nad nosilko vektorja b, točka S leži nad točko 0, torej v zgornji polravnini. Glede na lego sečišča trakov a_0 in b_{-1} (označimo ga z T) ločimo dva primera. V prvem, ko T leži v levi polravnini, območje $[2]^1_w$ omejujeta samo trak b_{-1} in ordinatna os, najnižja točka območja pa je točka S. V drugem primeru, ko T leži v desni polravnini, območje $[2]^1_w$ omejujejo ordinatna os, daljica $S-T$ in b_{-1} trak. Ker je trak b_{-1} naraščajoč, točka T leži nad točko S, točka S pa je ponovno najnižja točka območja $[2]^1_w$. Ker S leži nad ničlo, območje $[2]^1_w$ v celoti leži v zgornji polravnini.

Poiščimo zdaj transformacijo z zahtevanimi lastnostmi. Kot v drugih primerih pri $P=2$ bo tudi ta transformacija oblike $x \mapsto x+ta-sb$ pri $t,s \geq 0$. Elementom prvega kvadranta premiki v smeri a normo večajo, premiki v smeri $-b$ pa manjšajo, zato se zdi smiselno, da iskana transformacija ne bo vsebovala premika v smeri a, pač pa bo oblike $x \mapsto x-sb$. Toda žal tiči v takem razmišljanju napaka, saj transformacija te oblike v določenih primerih (to je, ko je x preblizu ordinatni osi), slika ven iz predpisanega območja (slika elementa x, za katerega velja $x_1 < sb_1$, je izven $(2)^1_w$), kar pa je v nasprotju s pogojem (a) izreka 3.13-III. Da se izognemo tej težavi, mora transformacija vsebovati vsaj en premik v smeri a. Ker pa se s takim premikom norma elementa poveča, je potrebno s primernim številom premikov v smeri $-b$ to povečanje norme popraviti. V nadaljevanju bomo dokazali, da je primerno število premikov enako $[a_1/b_1]$, oziroma da transformacija $\tau : x \mapsto x+a-[a_1/b_1]b$ izpolnjuje vse pogoje izreka.
Naj bo $k=[a_1/b_1]$. Ker $a_1 > b_1$ (glej posledico 3.10-V), je $k \geq 1$ in lahko napišemo

$$a_1 = kb_1 + z \text{ pri } 0 < z < b_1.$$

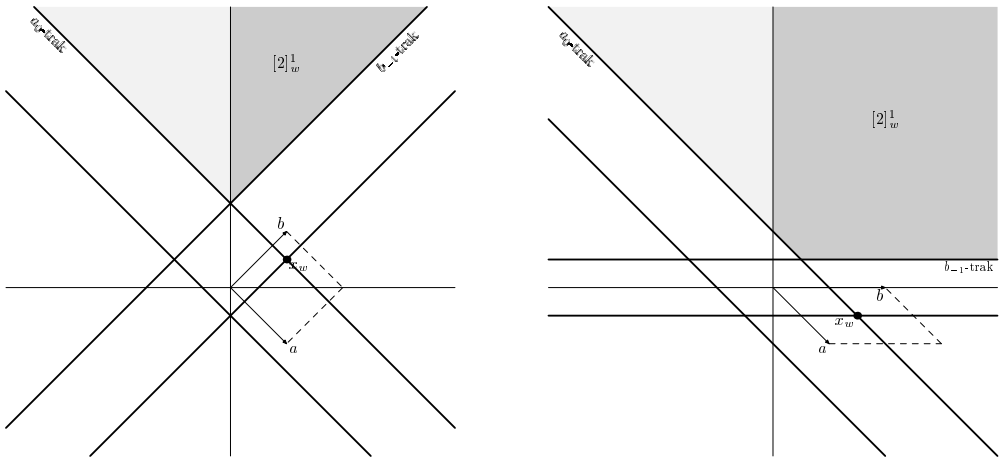
Vzemimo $x \in (2)^1_w$ in naj bo $y=\tau(x)$. Najprej se prepričajmo, da y leži v desni polravnini. Res, $y_1 = x_1 + a_1 - kb_1 = x_1 + kb_1 + z - kb_1 = x_1 + z > x_1$. Ker x leži v desni polravnini, tudi y leži tam. Zaradi oblike transformacije τ je slika elementov iz $(2)^1_w$ bodisi v $[2]_w$, bodisi v $[4]_w$. Toda, ker $[4]_w$ v celoti leži v tretjem kvadrantu (obe koordinati elementov tega prekata sta negativni), y ne more biti v $[4]_w$, saj smo ravnokar pokazali, da je $y_1 > 0$. Transformacija τ torej slika elemente iz $(2)^1_w$ v $[2]^1_w$ in izpolnjuje pogoj (a). Ker τ v vsakem primeru (ne glede na vrednost števila k) elemente približa traku a_0, od traku b_{-1} pa jih ne oddalji, je izpolnjen tudi pogoj (b). Da transformacija τ izpolnjuje tudi pogoj (c), se bomo prepričali tako, da bomo iz trditve $||y|| > ||x||$ izpeljali protislovje s predpostavko o pakiranosti baze. Ker sta x in y elementa prvega kvadranta, velja $||y|| = y_1 + y_2 = x_1 + z + x_2 + a_2 - kb_2$ in $||x|| = x_1 + x_2$. Iz $||y|| > ||x||$ potem sledi $x_1 + z + x_2 + a_2 - kb_2 > x_1 + x_2$ oziroma $a_2 > kb_2 - z$. Ker je $k \geq 1$ in $z < b$, je

$$a_2 > kb_2 - z > kb_2 - b_1 > b_2 - b_1$$

oziroma $-a_2 > b_1 - b_2$, in zato

$$||d|| = b_2 - a_2 + a_1 - b_1 < b_2 + b_1 - b_2 + a_1 - b_1 = a_1 < ||a||. \text{—}\star\text{—}$$

Ker je torej $||\tau(x)|| < ||x||$ za vsak $x \in (2)^1_w$, transformacija τ izpolnjuje tudi pogoj (c). □

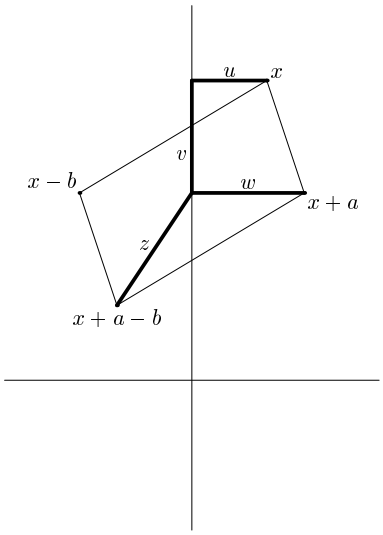


Slika 3.13-VII: Oblika in lega območja $[2]_w$ pri pakiranih bazah tipa F in H

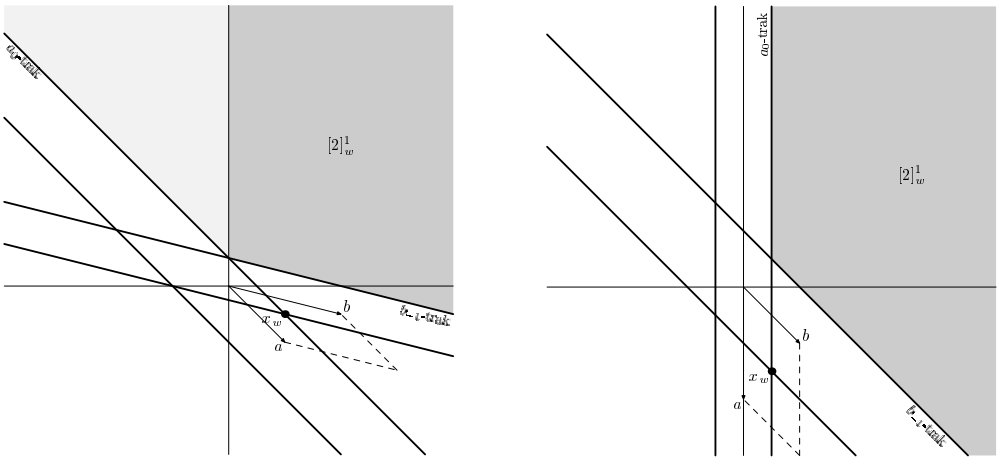
87 *Dokaz.* (trditev 3.13-III, $i = 1,\ P = 2$, bazi tipa F in H; glej sliko 3.13-VII)
Pri bazah tipa F in H območje $[2]_w$ v celoti leži v zgornji polravnini, o čemer se prepričamo podobno, kot smo to storili pri bazah tipa D in E. Glede na smer diagonale $d = a - b$ (po trditvah 3.10-VI in 3.10-VIII ima ta smer JV, J ali JZ) bomo ločili dva primera in za vsakega posebej poiskali primerno transformacijo.

(a) Smer diagonale d: JZ ali J
Pokazali bomo, da ena od transformacij $\tau_1 : x \mapsto x + a$ in $\tau_2 : x \mapsto x + d$ izpolnjuje vse tri pogoje. V primeru, da kot med absciso in vektorjem a ni manjši od 45° ($a_1 \leq |a_2|$), τ_1 izpolnjuje vse tri pogoje – da izpolnjuje (a) in (b), je razvidno iz definicije, (c) pa je izpolnjen, saj iz $a_1 \leq |a_2|$ neposredno sledi $||x + a|| \leq ||x||$. V nasprotnem primeru pa, ko je $a_1 > |a_2|$, oziroma $||x + a|| > ||x||$, pokažimo, da lahko uporabimo transformacijo τ_2. Prepričajmo se najprej, da τ_2 elemente iz $(2)^1_w$ slika v $[2]^1_w$. Iz oblike transformacije vidimo, da slika iz $(2)_w$ v $[2]_w$, prepričati se moramo le, da slika elementa iz $(2)^1_w$ ne leži v $[2]_w - (2)^1_w$, oziroma, da τ_2 elementov iz $(2)^1_w$ ne preslika preko ordinatne osi. Pa recimo, da bi bilo to res, oziroma, da obstaja $x \in (2)^1_w$, za katerega je $x + d \in [2]_w - (2)^1_w$. Potem bi (uporabljajoč oznake s slike 3.13-VIII) veljalo, da je norma diagonale d enaka $z + v + u$, norma vektorja b pa $z + w$. Ker pa je po privzetku norma elementa x manjša od norme elementa $x + a$, je $v + u < w$ in zato $||d|| = z + v + u < z + b = ||b||$, kar pa je v protislovju s predpostavko o pakiranosti baze. Transformacija τ_2 torej res slika iz $(2)^1_w$ v $[2]^1_w$, pogoj (a) je izpolnjen. Da τ_2 elementom prvega kvadranta manjša normo, je očitno, saj sta obe koordinati vektorja d nepozitivni. Ker τ_2 zmanjša obe količini, r_n in r_m, je izpolnjen tudi pogoj (c), oziroma, τ_2 ustreza iskani transformaciji.

(b) Smer diagonale d: JV
Za primer, pri katerem celotno območje $(2)^1_w$ leži v prvem kvadrantu in ima diagonala d



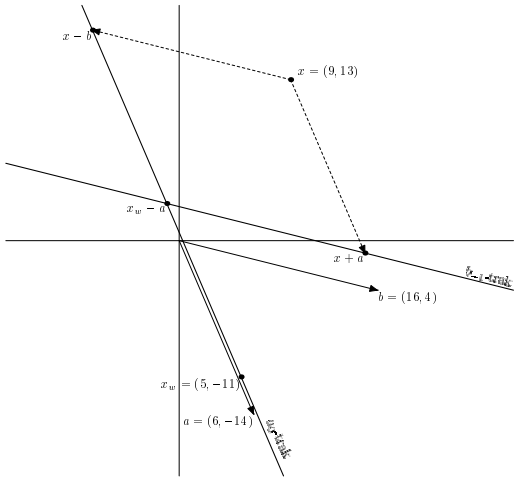
Slika 3.13-VIII: Protislovje: τ_2 ne more slikati preko ordinatne osi, saj je to v nasprotju s predpostavko o pakiranosti baze



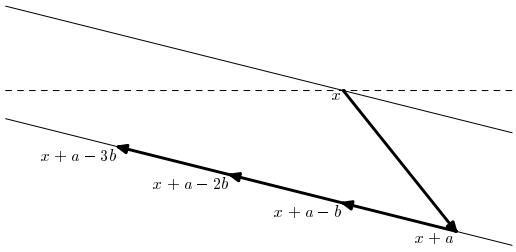
Slika 3.13-IX: Oblika in lega območja $[2]_w$ pri pakiranih bazah tipa J in K

smer JV, smo že dokazali (glej dokaz pri D in E), da transformacija $\tau : x \mapsto x + a - [a_1/b_1]b$ izpolnjuje vse tri pogoje izreka. □ ⌋86

Dokaz. (trditev 3.13-III, $i = 1,\ P = 2$, bazi tipa J in K; glej sliko 3.13-IX) ⌉90
Pri dokazovanju moramo upoštevati posebnost območja $[2]_w$ za bazi tipa J in K, namreč, da ta seka četrti kvadrant (pri bazah drugih tipov se to ni zgodilo); pri bazi tipa J območje $[2]_w$ seka celo tri kvadrante (vse razen tretjega). Pri iskanju primerne transformacije bomo morali biti zato posebno pozorni na elemente, ki bi jih ta preslikala preko abscisne ali



(a)

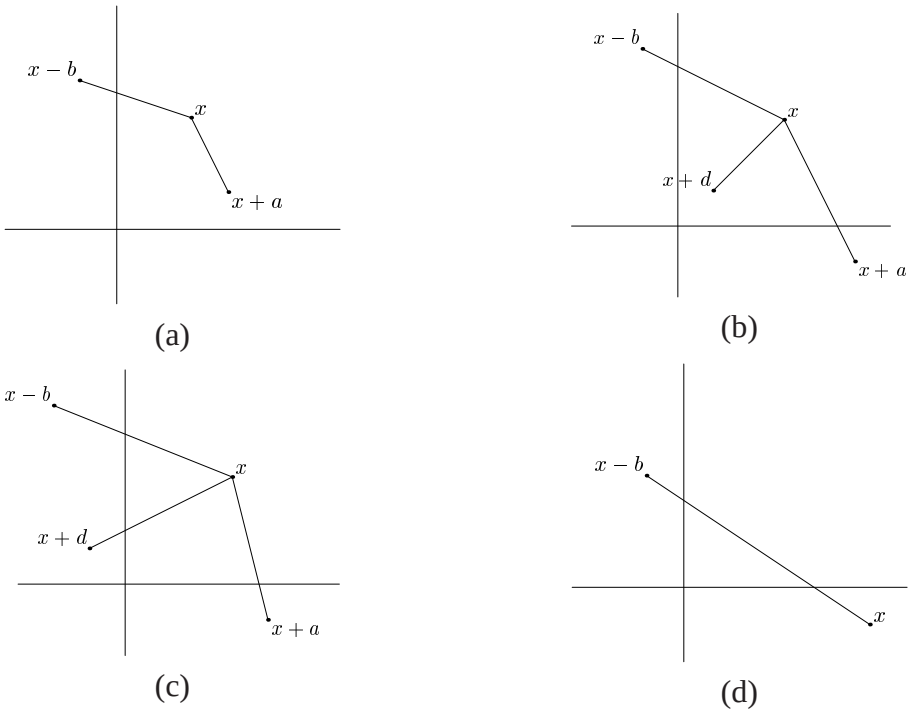


(b)

Slika 3.13-X: (a) primer pakirane baze $\{a, b\} = \{(6, -14), (16, 4)\}$ pri kateri je pri $x_w = (5, -11)$ za $x = (9, 13)$ iz prvega kvadranta slika $\tau(x)$ v drugem, $\tau_1(x)$ pa v četrtem kvadrantu in (b) shematski prikaz delovanja transformacije τ_3 na elementu četrtega kvadranta

ordinatne osi. Čeprav na prvi pogled morda kaže, da dokaz ne bo težaven, saj je transformacija $\tau : x \mapsto x - b$ dobra v skoraj vseh primerih, se izkaže, da je treba preučiti kar štiri posebne primere in za vsakega posebej dokazati primernost izbrane transformacije. Poglejmo najprej, v katerih primerih transformacija τ zadošča vsem pogojem. Ker τ za vsak $x \in (2)_w$ manjša tako vrednost števila $r_n(x)$ kot tudi normo, je primerna za vse x, katerih slika leži v predpisanem območju, to je v $[2]^1_w$. Če upoštevamo še, da τ po definiciji slika iz $(2)_w$ v $[2]_w$, lahko zaključimo, da je τ primerna transformacija za vse x, katerih slika ne leži v drugem kvadrantu. Ker območje $[2]_w$ pri bazi tipa K ne seka drugega kvadranta, je τ vedno primerna transformacija, torej je dokaz za tip K zaključen.
Za bazo tipa J pa poglejmo, katero transformacijo bi lahko uporabili v primeru, kadar τ odpove. Misel, da je v teh primerih dobra transformacija $\tau_1 : x \mapsto x + a$, se izkaže za neuporabno vsaj zaradi dveh razlogov: (1) če τ odpove na elementu prvega kvadranta, se v posebnih primerih lahko zgodi, da na tem elementu odpove tudi transformacija τ_1 (slika 3.13-X(a) prikazuje situacijo, ko τ slika preko ordinatne osi, τ_2 pa preko abscisne; ne moremo torej zagotovo trditi, da norma $||x + a||$ ni večja od norme $||x||$) in (2) τ lahko odpove tudi na elementih četrtega kvadranta (glej sliko 3.13-XI(d)), za katere τ_1 sploh ni primerna. Ker je, kot smo videli, več različnih primerov, pri katerih τ slika v četrti kvadrant, moramo obdelati vsakega posebej. Slika 3.13-XI prikazuje štiri mogoče postavitve elementov $x, x+a, x+d$ in $x-b$; na slikah a, b in c x leži v prvem, na sliki d pa v četrtem kvadrantu. Oglejmo si, katere transformacije lahko uporabimo v posameznem primeru.

Primer a. Ker sta tako x kot $x + a$ v prvem kvadrantu, je primerna transformacija oblike $x \mapsto x + a$. Res, (a) velja, saj je slika elementa $x + a$ desno od elementa x in zato



Slika 3.13-XI: Štiri postavitve elementov $x, x+a, x+d$ in $x-b$ pri pakirani bazi tipa J v primeru, ko $x-b$ leži v levi polravnini

gotovo v desni polravnini, (b) velja, ker transformacija te oblike zmanjša število $r_m(x)$ za ena, r_n pa ohrani, (c) pa je izpolnjen, ker je kot med vektorjem a in absciso dovolj velik (ni manjši od 45°).

Primer b. Če z d označimo diagonalo $d = a - b$, se lahko, podobno kot v primeru a, tudi v danem primeru prepričamo, da je zaradi dejstva, da tako x kot $x + d$ ležita v prvem kvadrantu, primerna transformacija oblike $x \mapsto x + d$. Omenimo še, da je za ta primer primerna tudi transformacija $x \mapsto x + a$, kar dokažemo enako kot v primeru c (glej spodaj), saj $x + d$ v obeh primerih leži v zgornji polravnini (na tem dejstvu namreč temelji dokaz v primeru c).

Primer c. Primerna transformacija za ta primer bo oblike $\tau_1 : x \mapsto x + a$. V to se prepričamo tako, da pokažemo, da norma $||x + a||$ ni večja od norme $||x||$ (to bi se, v primeru, ko baza ne bi bila pakirana, lahko zgodilo, saj $x + a$ leži na drugi strani abscisne osi). Ker $x + d$ leži v levi polravnini, gotovo leži v tretjem kvadrantu, saj območje $[2]_w$ ne seka četrtega kvadranta, torej je $(x + d)_2 \geq 0$. Od tod pa sledi $x_2 \geq b_2 - a_2$ oziroma $x_2 \geq |a_2| - |b_2|$ (res, saj sta tako a_2 kot b_2 negativni števili in velja $|a_2| > |b_2|$). Za bazne vektorje pakirane baze četrtega kvadranta iz $||a - b|| = b_1 - a_1 + |a_2| - |b_2|$ in iz $||a - b|| \geq ||b||$ sledi $-|b_2| \geq (a_1 - |a_2|)/2$, zato je $x_2 \geq |a_2| + (a_1 - |a_2|)/2 = ||a||/2$. Oglejmo si velikost norme $||x + a||$.

$$||x + a|| = |x_1 + a_1| + |x_2 + a_2| = x_1 + a_1 + |a_2| - x_2 = ||a|| + x_1 - x_2.$$

Ker je $||a|| \leq 2x_2$, velja

$$||x + a|| \leq 2x_2 + x_1 - x_2 = x_1 + x_2 = ||x||.$$

Primer d. Podobno kot pri dokazu za primer pakirane baze tipa D in E, bo tudi v tem primeru primerna transformacija oblike $\tau_2 : x \mapsto x + a - kb$ pri $k = [a_2/b_2]$. Neformalno to izbiro utemeljimo takole: ker premik elementa x v smeri $-b$ slika izven dovoljenega območja, moramo uporabiti en premik v smeri a. Tak premik pa normo poveča, zato je treba s primernim številom premikov v smeri $-b$ poskrbeti, da se bo ta vsaj za toliko tudi zmanjšala. Število premikov v smeri $-b$ izberemo tako, da je končna slika še vedno v četrtem kvadrantu, norma slike pa ni večja od norme začetnega elementa. Shematsko je pravkar opisani premik prikazan na sliki 3.13-X(b), formalno pa primernost izbrane transformacije utemeljimo, kot sledi. Ker je $|a_2| > |b_2|$, pišimo $a_2 = kb_2 + z$ pri $b_2 < z < 0$. Naj bo x element četrtega kvadranta, za katerega velja $x - b \in \mathbb{Z}^2_{-1}$, in naj bo y slika elementa x, to je $y = \tau_2(x)$. Ker je $y_2 = x_2 + a_2 - kb_2$, je $y_2 < x_2$, torej y leži v spodnji polravnini, oziroma v četrtem kvadrantu, saj območje $[2]_w$ v tem primeru tretjega kvadranta ne seka. Slika elementa x je v predpisanem območju, zato je pogoj (a) izpolnjen. Transformacija τ_2 zmanjša $r_n(x)$ in $r_m(x)$, zato je tudi pogoj (b) izpolnjen. Poglejmo še, kako je z normo elementa y. Ker je

$$\begin{aligned} |y_1| &= x_1 + a_1 - kb_1 \\ |y_2| &= |x_2 + a_2 - kb_2| = |x_2 + kb_2 + z - kb_2| = |x_2 + z| = |x_2| + |z|, \end{aligned}$$

je

$$||y|| - ||x|| = x_1 + a_1 - kb_1 + |x_2| + |z| - x_1 - |x_2| = a_1 - kb_1 + |z|.$$

Recimo, da je norma $||y||$ večja od norme $||x||$, oziroma, $a_1 - kb_1 + |z| > 0$. Potem je

$$a_1 > kb_1 - |z| > kb_1 - |b_2| > b_1 - |b_2|,$$

oziroma $-a_1 < |b_2| - b_1$. Za normo diagonale $d = a - b$ torej velja

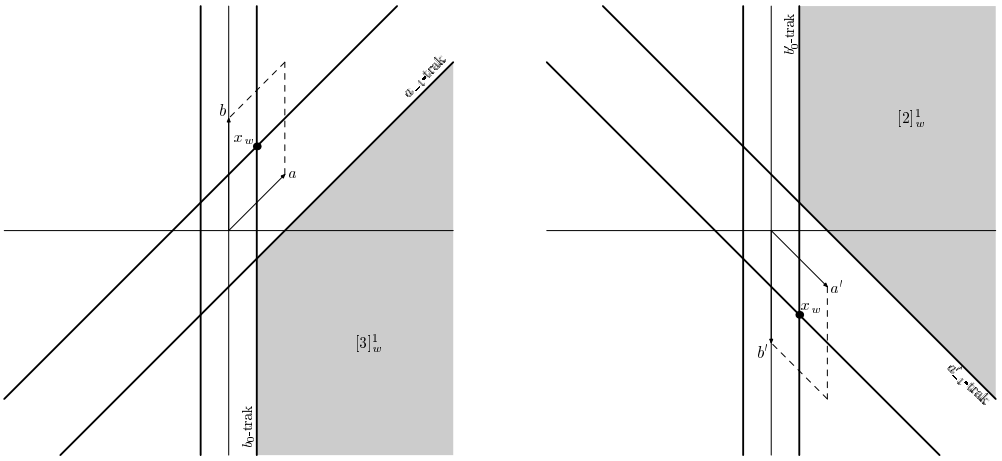
$$||d|| = b_1 - a_1 + |a_2| - |b_2| < b_1 + |b_2| - b_1 + |a_2| - |b_2| < ||a||,$$

kar pa je v nasprotju s predpostavko o pakirani bazi. Torej je $||y|| \leq ||x||$, oziroma transformacija τ_2 izpolnjuje tudi pogoj (c).

87 ⌊ □

Dokaz trditve 3.13-III za ostale primere ↓

91 ⌈ *Dokaz.* (trditev 3.13-III, $i = 1,\ P = 3$)
Dokaz za primer $P = 3$ bi lahko predstavili v enaki obliki, kot smo predstavili dokaz za primer $P = 2$: za vsak tip pakirane baze bi poiskali primerno transformacijo in dokazali,



(a) (b)

Slika 3.13-XII: (a) Oblika in lega območja $[3]_w$ za pakirano bazo tipa A ter (b) zrcalna slika tega območja (zrcaljenje preko abscisne osi)

da ta res izpolnjuje vse tri pogoje. Tak pristop bi bil sicer formalno pravilen, vendar utrujajoč, saj ne bi prinesel nič novega. Izkaže se namreč, da je v dokazu za primer $P = 2$ v celoti zajet tudi dokaz za $P = 3$. Poglejmo si pomen te trditve na primeru. Vzemimo pakirano bazo tipa A, narišimo območje $[3]^1_w$ (slika 3.13-XII(a)) in ga prezrcalimo preko abscisne osi (slika 3.13-XII(b)). Dobili smo znano sliko – pakirana baza $\{a, b\}$ tipa A se je preslikala v pakirano bazo $\{b, a\}$ tipa K, območje $[3]^1_w$ pa v območje $[2]^1_w$. Za ta primer pa smo že dokazali, da je iskana transformacija oblike $x \mapsto x-b$, zato bo za primer $P = 3$ pri bazi tipa A iskana transformacija oblike $x \mapsto x - a$ (pri preslikavi preko abscise se je zamenjal pomen vektorjev a in b – vektor, ki leži spodaj, je označen z b – zato smo črki a in b zamenjali tudi pri transformaciji). Pravkar opisan postopek lahko uporabimo pri vseh tipih baze in sicer takole: za izbrani tip baze X določimo tip preslikane baze X'; transformacijo, ki bo primerna za primer $P = 3$ pri pakirani bazi tipa X, dobimo tako, da v transformaciji za primer $P = 2$ pri pakirani bazi tipa X' zamenjamo črki a in b. Primernost tako izbrane transformacije opravičimo s pravkar podanim razmislekom in dokazom za odgovarjajoči tip baze pri $P = 2$.
Za konec dokaza podajmo še permutacijo π, iz katere je razvidno, kako se pri preslikavi preko abscisne osi spremeni tip pakirane baze:

$$\pi = \begin{pmatrix} A & B & C & D & E & F & G & H & I & J & K \\ K & I & G & J & H & F & C & E & B & D & A \end{pmatrix},$$

in pa tabelo 3.13-III, ki prikazuje primerne transformacije za posamezen tip baze pri $P = 3$. □

Dokaz. (trditev 3.13-III, $i = 1,\ P = 4$)
Dokaz za $P = 4$ je preprost. Velja namreč, da ne glede na tip pakirane baze območje $[4]_w$ leži v levi polravnini. Res, za $x \in [4]_w$ velja $x = x_w - ta - sb$ pri $t, s > 0$. Ker je

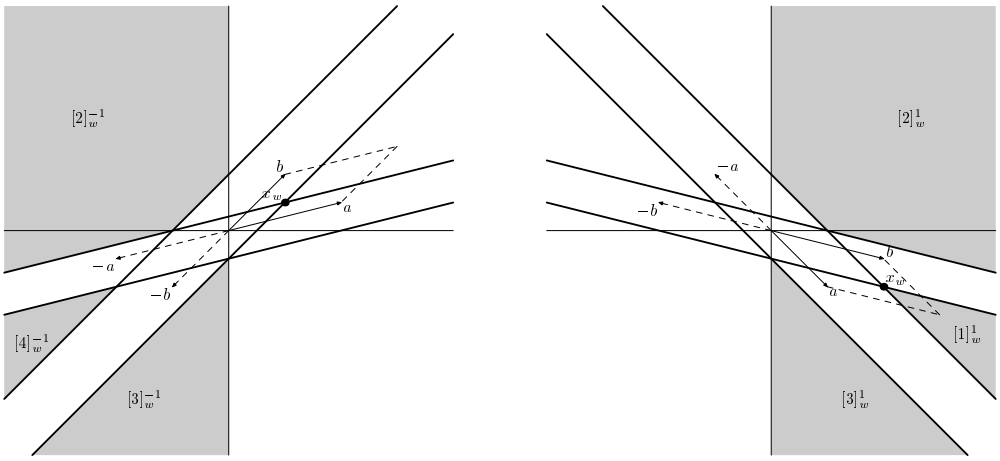
tip pakirane baze	primerna transformacija
A, B, C	$x \mapsto x - a$
D	1. $x - a \in \mathbb{Z}_1^2$: $x \mapsto x - a$ 2. sicer (a) x leži v 4. kvadrantu: $x \mapsto x + b$ (b) x leži v 1. kvadrantu: $x \mapsto x - [b_2/a_2]a + b$
E	$x \mapsto x + b$
F	1. $a_1 \geq b_1$ (a) $b_2 \geq b_1$: $x \mapsto x + b$ (b) sicer: $x \mapsto x + b - a$ 2. sicer : $x \mapsto x - [b_1/a_1]a + b$
G, I, K	$[3]_w \cap \mathbb{Z}_1^2 = \emptyset \Rightarrow$ vsaka transformacija zadošča pogojem
H, J	$x \mapsto x - [b_1/a_1]a + b$

Tabela 3.13-III: Primerne transformacije za posamezen tip baze za primer $i = 1,\ P = 3$

$x_{w1} < a_1 + b_1$, je $x_1 < 0$. Torej je $[4]_w^1 = \emptyset$ in vsi pogoji izreka so na prazno izpolnjeni. □

Dokaz. (trditev 3.13-III, $i = -1$)
V tem delu dokaza bomo uporabili podobne utemeljitve, kot smo to storili pri $i = 1,\ P = 3$, le da bomo tokrat uporabili zrcaljenje preko ordinatne osi. Tako zrcaljenje nam območje $[2]_w^{-1}$ pakirane baze $\{a, b\}$ tipa X preslika v območje $[2]_w^1$ pakirane baze $\{-b, -a\}$ tipa $\pi(X)$ (permutacija π je definirana v dokazu za primer $i = 1,\ P = 3$), območje $[3]_w^{-1}$ v $[3]_w^1$, območje $[1]_w^{-1}$ v $[4]_w^1$ in območje $[4]_w^{-1}$ v $[1]_w^1$ (glej sliko 3.13-XIII). Zato lahko dokaz zaključimo takole: za vsako območje P pri $i = -1$ in pakirani bazi $\{a, b\}$ tipa X je iskana transformacija enaka transformaciji, ki je primerna za območje P' pri $i = 1$ in pakirani bazi $\{-b, -a\}$ tipa $\pi(X)$ ($P' = P$ za $P = 2, 3$ in $P' = 5 - P$ za $P = 1, 4$). □

Dokaz. (trditev 3.13-III, $i = 2$ in $i = -2$)
Trditev 3.13-III pri $i = 2$ in $i = -2$ zajema 88 posebnih primerov (vsaka urejena trojica $(i, P, B) \in \{2, -2\} \times \{1, 2, 3, 4\} \times \{A,B,C,D,E,F,G,H,I,J,K\}$ predstavlja enega izmed njih). Med njimi je 30 takih, da je množica $[P]_w^i$ prazna, torej je izrek na prazno izpolnjen, za preostalih 58 pa moramo za vsakega posebej poiskati transformacijo, ki zadošča vsem trem pogojem izreka. K sreči pa lahko to težavno opravilo poenostavimo. Namreč, podobno kot pri dokazovanju trditve za primer $i = -1$, lahko v vsakem primeru pri $i = 2$ in $i = -2$ vidimo (če nanj pogledamo iz pravega zornega kota), že znan in dokazan primer. Pri $i = -1$ smo morali, če smo želeli sliko iz leve polravnine preslikati v desno in na ta način priti do že znane situacije, uporabiti zrcaljenje preko ordinatne osi,

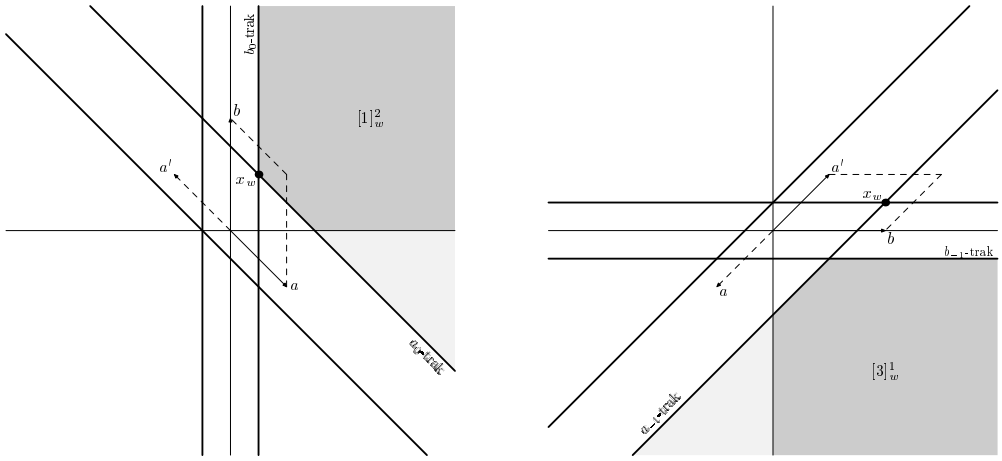


(a) (b)

Slika 3.13-XIII: (a) Oblika in lega območij $[1]_w^{-1}, [2]_w^{-1}, [3]_w^{-1}$ in $[4]_w^{-1}$ za pakirano bazo $\{a, b\}$ tipa D ter (b) slika (a), zrcaljena preko ordinatne osi: območja $[1]_w^1, [2]_w^1, [3]_w^1$ in $[4]_w^1$ za pakirano bazo $\{-b, -a\}$ tipa J

v primerih $i = 2$ in $i = -2$ pa moramo zgornjo oziroma spodnjo polravnino preslikati v levo polravnino, zato bomo uporabili rotacijo za 90° (v prvem primeru v negativni, v drugem pa v pozitivni smeri). Pri rotaciji se nam bo morda zgodilo, da se bo kateri od baznih vektorjev preslikal ven iz prvega oziroma četrtega kvadranta – namesto takega vektorja bomo uporabili obratno usmerjeni vektor (torej, $-a$ namesto a in $-b$ namesto b). Po rotaciji in morebitnem popravku vektorjev dobimo enega od znanih in dokazanih primerov – transformacija, ki je primerna za ta znani primer, bo primerna tudi za primer, ki ga dokazujemo. V nadaljevanju dokaza si bomo natančneje ogledali dokaz za enega od omenjenih 58 primerov – preostalih 57 lahko bralec po istem postopku dokaže za vajo.

Slika 3.13-XIV(a) prikazuje območje $[1]_w^2$ za pakirano bazo tipa C. Na sliki je označena baza $\{a, b\}$ in vektorja $-a$ ter $-b$. Po rotaciji te slike za 90° v smeri urinega kazalca dobimo sliko, ki prikazuje območje $[3]_w^1$ za pakirano bazo $\{a', b'\} = \{b, -a\}$ tipa E. Iz dokaza pri $i = 1,\ P = 3$ vemo, da je v primeru pakirane baze tipa E primerna transformacija oblike $x \mapsto x + b'$, torej lahko zaključimo, da je za primer $i = 2,\ P = 1$ pri bazi tipa C primerna transformacija oblike $\tau : x \mapsto x - a$. □



(a) (b)

Slika 3.13-XIV: (a) Oblika in lega območja $[1]^2_w$ pri pakirani bazi $\{a, b\}$ tipa C in (b) območje $[3]^1_w$ pri bazi $\{b, -a\}$ tipa E (slika (b) je za 90° zarotirana slika (a))

Povzetek

Dokaz trditve 3.13-III je tako končan. Videli smo, da je dovolj, če poznamo primerne transformacije za primera $i = 1, P = 1$ ter $i = 1,\ P = 2$, saj lahko vse druge primere prevedemo nanju. Ponovimo torej, kakšne so transformacije za omenjena primera. Primerna transformacija za primer $i = 1,\ P = 1$ je bodisi

$$x \mapsto x - a$$

bodisi

$$x \mapsto x - b,$$

saj za vsak x vsaj ena od njiju izpolnjuje vse pogoje. Za primer $i = 1,\ P = 2$ pa glede na tip pakirane baze ter glede na medsebojno lego baznih vektorjev ločimo naslednje primere:

baze tipa A, B in C:
množica $[2]^1_w$ je prazna,

baze tipa D, E, F, G, H in I:

1. smer diagonale $d = a - b$ je JZ ali J (G, H, I in F):
   - $|a_1| \leq |a_2| \rightarrow \tau : x \mapsto x + a$,
   - $|a_1| > |a_2| \rightarrow \tau : x \mapsto x + d$,
2. smer diagonale $d = a - b$ je JV (D, E in F):
   $\rightarrow \tau : x \mapsto x + a - [a_1/b_1]b$,

bazi tipa J in K:

1. $x - b \in \mathbb{Z}_1^2 \rightarrow \tau : x \mapsto x - b$,
2. $x - b \in \mathbb{Z}_{-1}^2$:
   - x leži v 1. kvadrantu $\rightarrow \tau : x \mapsto x + a$,
   - x leži v 4. kvadrantu $\rightarrow \tau : x \mapsto x + a - [a_2/b_2]b$.

Poglavje 4

Paketno usmerjanje v omrežjih s topologijo krožnih grafov

Usmerjanje podatkov, ki je osnovna funkcija vsakega računalniškega omrežja [36], zajema širok spekter problemov, od katerih se bomo v tem poglavju posvetili predvsem osnovnemu usmerjanju in splošnemu usmerjevalnemu problemu. Osredotočili se bomo na omrežja s topologijo krožnih grafov in predstavili pet statičnih in en dinamičen usmerjevalni algoritem za neusmerjene dvokrožne grafe ter posplošitev dinamičnega algoritma za primer splošnega neusmerjenega k-krožnega grafa.

Od problema k algoritmu
Naloga algoritma, ki rešuje problem osnovnega usmerjanja, je, da izbere eno od najkrajših poti med izvorom in ponorom paketa ter nato paket po izbrani poti pripelje do cilja. Ker je pri tem problemu v omrežju vedno samo en paket, do trkov med paketi ne more priti, skupno število usmerjevalnih korakov pa je enako dolžini najkrajše poti. Žal ta lepa lastnost velja le za problem osnovnega usmerjanja. Že v primeru, ko imamo v omrežju dva paketa, se namreč lahko zgodi, da trka ne moremo preprečiti. Pri reševanju splošnega usmerjevalnega problema pa je položaj še težji. Zaradi velikega števila paketov v omrežju so pri reševanju tega problema trki težava, ki zahteva resno obravnavo. Algoritem za reševanje splošnega usmerjevalnega problema lahko razumemo kot nadgradnjo algoritma za osnovno usmerjanje, saj mora imeti poleg sposobnosti izbiranja najkrajše poti vgrajen tudi mehanizem za preprečevanje in reševanje posledic trkov.

Idealen usmerjevalni algoritem trke rešuje na “globalni ravni” – ob poznavanju podatkov o celotnem problemu poišče najboljšo pot za vsak paket (to je najkrajšo pot, na kateri bo srečal najmanj paketov). Tako problem reši na optimalen način. Trkom, ki pri takem usmerjanju nastanejo, se ne da ogniti, skupno število usmerjevalnih korakov pa je enako minimalnemu številu usmerjevalnih korakov, ki so potrebni za rešitev danega problema. Tako reševanje problemov je s teoretičnega stališča sicer zanimivo, s praktičnega pa skoraj povsem neuporabno. Ne samo, da je predpostavka o globalnem poznavanju celotnega problema nerealna, tudi v primeru, ko poznamo vse podatke, je iskanje optimalne rešitve

zahtevna naloga. V splošnem primeru je treba pregledati vse možnosti, kar pa je pri velikem številu paketov in/ali vozlišč omrežja zamudno opravilo. Zato se pri našem delu z optimalnimi algoritmi za splošen usmerjevalni problem nismo ukvarjali. Namesto tega smo že v prejšnjih raziskavah razvili in ovrednotili statični usmerjevalni algoritem $\mathcal{S}_4$, ki trke rešuje po hevristični metodi. Omenjeni algoritem smo razvili postopoma. Začeli smo s preprostim statičnim usmerjevalnim algoritmom $\mathcal{S}_0$ – metoda za reševanje trkov pri tem algoritmu temelji na preprostih predpostavkah. S pomočjo obsežnih simulacij delovanja algoritma $\mathcal{S}_0$ smo najprej razvili algoritem $\mathcal{S}_1$, nato pa, spet na podlagi rezultatov simulacij, še algoritme $\mathcal{S}_2$, $\mathcal{S}_3$ in $\mathcal{S}_4$. Vsaka naslednja različica algoritma je izboljšava prejšnje. Algoritem $\mathcal{S}_4$ se, glede na rezultate naših simulacij, zelo približa teoretično optimalnemu usmerjevalnemu algoritmu.

Model usmerjevalnega algoritma

Ko smo pri razvoju $\mathcal{S}_0$, $\mathcal{S}_1$, $\mathcal{S}_2$, $\mathcal{S}_3$ in $\mathcal{S}_4$ algoritme implementirali v simulatorju, pa tudi kasneje, ko smo jih poskušali lepo opisati, se je pokazala potreba po enotnem zapisu usmerjevalnih algoritmov. Ne samo, da bi s pomočjo takega zapisa algoritme lažje implementirali, algoritmi, zapisani na enoten način, so med seboj tudi bolj primerljivi. Zato smo v okviru dela doktorske disertacije izdelali t.i. *model usmerjevalnega algoritma*, ki táko enotno predstavitev omogoča. Algoritem v omenjenem modelu podamo tako, da opišemo

a. Način preprečevanja in reševanja posledic trkov
Model omogoča več različnih a) vrstnih redov izbire povezav, b) možnosti izbire paketov v čakalni vrsti posameznega vozlišča, c) načinov določanja prioritete potovanja posameznega paketa in d) načinov vstavljanja poraženih paketov nazaj v čakalno vrsto. Vse omenjene opcije opredeljujejo način preprečevanja in reševanja posledic trkov. Posamezno opcijo določimo z nastavitvijo parametra modela. Način preprečevanja in reševanja posledic trkov danega algoritma v modelu podamo tako, da podamo odgovarjajoč *nabor parametrov* za preprečevanje in reševanje posledic trkov.

b. Način, s katerim algoritem posameznemu paketu zagotavlja, da ta ostane na najkrajši poti do cilja
Ker vrstni red povezav na poti paketa ne vpliva na njegov končni cilj (glej poglavje 3.5 o poteh v krožnih grafih), ima posamezni algoritem možnost izbire, po kateri od ekvivalentnih poti bo neki paket potoval. Algoritmu lasten način določanja poti v modelu delno opišemo z naborom parametrov, delno pa z implementacijo funkcije NASLEDNJA_POVEZAVA(p, j), ki odgovori na vprašanje, ali je povezava tipa j na poti paketa p; natančneje, funkcija NASLEDNJA_POVEZAVA mora biti implementirana tako, da s True odgovori le, če je na poti paketa p zagotovo še kakšna povezava tipa j in s False, če take povezave ni, oziroma algoritem tega ne zna določiti.

Algoritem je torej v modelu natančno določen z naborom parametrov za preprečevanje in reševanje posledic trkov ter z načinom implementacije funkcije NASLEDNJA_POVEZAVA.

Zapis statičnih algoritmov v modelu
Statični algoritmi $\mathcal{S}_i$ ($i = 0, 1, 2, 3, 4$) uporabljajo dve različni metodi NASLEDNJA_POVEZAVA (NASLEDNJA_POVEZAVA$_{\mathcal{BF}}$ in NASLEDNJA_POVEZAVA$_{\mathcal{SB}}$). Metoda NASLEDNJA_POVEZAVA$_{\mathcal{BF}}$, uporabljena v prvi dveh algoritmih, je omejevala uporabo kratkih povezav in se je izkazala za neučinkovito, zato smo jo v naslednjih različicah nadomestili z drugo, NASLEDNJA_POVEZAVA$_{\mathcal{SB}}$. Čeprav sta omenjeni metodi različni, sta v zasnovi zelo podobni – obe rezultat vračata na podlagi vnaprej pripravljenih podatkov. Statični algoritmi $\mathcal{S}_0$, $\mathcal{S}_1$, $\mathcal{S}_2$, $\mathcal{S}_3$ in $\mathcal{S}_4$ namreč v fazi predprocesiranja določijo ekvivalenčni razred poti za posamezen paket; metodi NASLEDNJA_POVEZAVA nato, vsaka na svoj način, v konstantnem času izbereta povezavo na konkretni poti tega razreda. Bistveno različno metodo NASLEDNJA_POVEZAVA bomo srečali šele pri dinamičnem algoritmu $\mathcal{D}$ (glej spodaj), ki ekvivalenčni razred in konkretno pot paketa izračuna sproti na posameznem usmerjevalnem koraku.
Pet različnih statičnih algoritmov enačimo s petimi nabori parametrov usmerjevalnega modela. Na podlagi rezultatov simulacij lahko ugotovimo, kako nastavitev posameznega parametra vpliva na kvaliteto algoritma.

Dinamičen algoritem
V okviru dela doktorske disertacije smo razvili učinkovit algoritem, s pomočjo katerega lahko implementiramo funkcijo NASLEDNJA_POVEZAVA na dinamičen način. Algoritem, ki smo ga poimenovali z $\mathcal{RSP}$, v logaritemskem času določi množico vseh kandidatov za naslednjo povezavo na najkrajši poti danega paketa. Prednost takega pristopa je v tem, da za izračun nadaljevanja poti ne potrebuje nobenih vnaprej pripravljenih podatkov, poleg tega pa je bolj prilagodljiv, saj omogoča prehajanje med ekvivalenčnimi razredi najkrajših poti in je zato bolj odporen za napake.
Z uporabo algoritma $\mathcal{RSP}$ smo razvili dinamičen usmerjevalni algoritem za splošno usmerjanje $\mathcal{D}$. Ker za reševanje trkov algoritem $\mathcal{D}$ uporablja isti nabor parametrov kot algoritem $\mathcal{S}_4$, lahko na podlagi primerjave kakovosti obeh algoritmov ugotovimo, katera metoda za določanje nadaljevanja poti je ustreznejša.

Osnovno in splošno usmerjanje
Osnovno usmerjanje je poseben, najlažji primer splošnega usmerjanja, pri katerem je v omrežju samo en paket. Do trkov v tem primeru na more priti, zato je delovanje algoritma neodvisno od nastavitve parametrov modela. Najpomembnejši del algoritma za osnovno usmerjanje je torej funkcija NASLEDNJA_POVEZAVA – na podlagi rezultatov te funkcije algoritem izbere eno izmed povezav za nadaljevanje poti in paket pošlje po njej. Pošiljanje vedno uspe. Pri algoritmih za splošno usmerjanje je sama funkcija NASLEDNJA_POVEZAVA premalo – za uspešno delovanje algoritma je potrebna še strategija reševanja trkov, ki je določena z naborom parametrov. Algoritem za splošno usmerjanje lahko torej razumemo tudi kot nadgradnjo pripadajočega algoritma za osnovno usmerjanje.

Usmerjevalne omejitve
V nadaljevanju tega poglavja predpostavljamo neusmerjen, večsmeren shrani-in-pošlji pa-

ketni model usmerjanja podatkov (*angl. undirected multiport store-and-forward packet routing model*). Podatki v tem modelu se po omrežju prenašajo v obliki paketov. Ti lahko potujejo po povezavah v obeh smereh, toda ne hkrati (v vsakem usmerjevalnem koraku lahko povezava prenaša podatke le v eni smeri). Vozlišča lahko v vsakem koraku sprejmejo ali pošljejo nič ali več paketov, skupno število poslanih in prejetih paketov pa je kvečjemu enako stopnji posameznega vozlišča. Paketi od začetnega vozlišča potujejo po omrežju preko vmesnih vozlišč do svojega cilja. Pri tem se lahko v vsakem usmerjevalnem koraku premaknejo iz vozlišča, v katerem trenutno so, v eno od sosednjih vozlišč. V posameznem vozlišču lahko v čakalni vrsti čaka tudi več kot en paket. Velikosti čakalnih vrst nismo omejevali.

O poglavju
V prvem delu tega poglavja predstavimo usmerjevalne algoritme za dvokrožne grafe, v drugem pa uporabo teh algoritmov v splošnih k-krožnih grafih. Ker je osnovno usmerjanje uporabno pri reševanju splošnega usmerjevalnega problema, najprej predstavimo algoritma za statično in dinamično osnovno usmerjanje. Nato opišemo model usmerjevalnega algoritma in definiramo parametre, s katerimi posamezen algoritem v modelu opišemo. Sledita razdelek, v katerem predstavimo pet statičnih usmerjevalnih algoritmov $\mathcal{S}_i$ ($i = 0, 1, 2, 3, 4$), ter razdelek, v katerem te algoritme zapišemo s parametri omenjenega modela. Predstavimo tudi nov dinamični usmerjevalni algoritem $\mathcal{D}$ in podamo primerjavo med njim in statičnim algoritmom $\mathcal{S}_4$. V zadnjem razdelku opišemo posplošitev predstavljenega dinamičnega algoritma za uporabo v splošnih k-krožnih grafih.

4.1 Osnovno usmerjanje

Osnovno usmerjanje je poseben primer splošnega usmerjevalnega problema, pri katerem je v omrežju en sam paket. Ker do trkov pri reševanju tega problema ne more priti, se naloga usmerjevalnega algoritma poenostavi na iskanje najkrajše poti med izvorom in ponorom omenjenega paketa. Srce usmerjevalnega algoritma za osnovno usmerjanje je funkcija NASLEDNJA_POVEZAVA, s pomočjo katere algoritem na posameznem usmerjevalnem koraku sestavi množico povezav, ki so del najkrajše poti do cilja. Od tipa usmerjevalnega algoritma je odvisno, kako je funkcija NASLEDNJA_POVEZAVA implementirana – medtem, ko statični algoritmi v fazi predprocesiranja pripravijo vse potrebno, da funkcija NASLEDNJA_POVEZAVA rezultat vrača v konstantnem času, dinamični algoritmi vse podatke za izračun tega rezultata pripravijo sproti. V obeh primerih pa velja, da paket, ki ga algoritem pošlje po eni izmed povezav, za katere NASLEDNJA_POVEZAVA vrne True, ves čas ostaja na najkrajši poti do cilja. Postopek za reševanje osnovnega usmerjevalnega problema z uporabo funkcije NASLEDNJA_POVEZAVA je prikazan v algoritmu 4.1-I.
V podpoglavjih 4.1.1 in 4.1.2, ki sledita, predstavimo dve različni metodi za izračun funkcije NASLEDNJA_POVEZAVA: prva, NASLEDNJA_POVEZAVA$_{\mathcal{SB}}$ [55], rezultat vrne na statičen, druga, $\mathcal{RSP}$ [27], pa na dinamičen način.

Algoritem 4.1-I

Osnovno usmerjanje paketa p

```
1: while (paket p ni na cilju) do begin
2:     C:={};
3:     for i ∈ I do
4:         if NASLEDNJA_POVEZAVA(p, i) then C := C ∪ {E_i};
5:     p pošlji po eni izmed povezav množice C
6: end.
```

4.1.1 Statičen algoritem

Osnovna zamisel statičnega algoritma $\mathcal{SB}$ je naslednja: v času predprocesiranja, torej pred dejanskim začetkom usmerjanja, algoritem izbere ekvivalenčni razred sprehodov med začetnim in končnim vozliščem danega paketa; v času usmerjanja, to je na posameznem usmerjevalnem koraku, paket vodi proti cilju po enem od sprehodov, ki pripadajo izbranemu ekvivalenčnemu razredu. Kljub statični naravi opisanega algoritma ostaja nekaj svobode pri določanju konkretnega sprehoda, vendar je ta svoboda omejena na izbrani ekvivalenčni razred. Algoritem $\mathcal{SB}$ namreč nima sposobnosti preklapljanja med razredi niti v primeru, ko bi bilo to, zaradi morebitne okvare omrežja ali zasedenosti katere od povezav, nujno potrebno.

Algoritem $\mathcal{SB}$ v času predprocesiranja za paket, ki je v začetnem vozlišču u in je namenjen v vozlišče v, določi ekvivalenčni razred sprehodov tako, da izračuna tak par $(s, b) \in \mathbb{Z}_2^2$, ki med vsemi rešitvami enačbe $u + sh_1 + bh_2 \equiv v \pmod n$ minimizira $|s| + |b|$ (kako tako rešitev poiščemo v logaritemskem času, je opisano v poglavju 3.11 na strani 69). Par (s, b) nato pripne paketu in tako omogoči, da bo podatek o izbranem ekvivalenčnem razredu dosegljiv vsakemu vozlišču, kjer se bo paket med svojim potovanjem ustavil. Algoritem na posameznem usmerjevalnem koraku poskrbi, da se vrednost pripetega števila s oziroma b spremeni v skladu z izbrano povezavo (pred pošiljanjem paketa po kratki (oz. dolgi) povezavi naprej zmanjša vrednost števila s (oz. b); podobno stori pri pošiljanju po povezavah v smeri nazaj, le da tokrat števili s oz. b poveča). Paket doseže svoj cilj, ko pripeti števili s in b dosežeta vrednost 0.

> Primer 4.1-I. V grafu $G(10;\ \pm 3, \pm 4)$ imamo paket, ki je iz vozlišča 2 namenjen v vozlišče 7. Najkrajša pot med omenjenima vozliščema je sestavljena iz kratke povezave v smeri nazaj in iz dveh dolgih v smeri naprej ($2 - 3 + 4 + 4 \equiv 7 \pmod{10}$), zato algoritem $\mathcal{SB}$ paketu v času predprocesiranja pripne par $(s, b) = (-1, 2)$. V prvem koraku, ko paket pošlje po kratki povezavi v smeri nazaj iz vozlišča 2 v vozlišče 9, $\mathcal{SB}$ poveča vrednost števila s, v drugem in tretjem, ko uporabi dolge povezave v smeri naprej, pa zmanjša vrednost števila b. Po treh korakih paket doseže cilj, števili s in b pa vrednost 0.

Algoritem 4.1-II NASLEDNJA_POVEZAVA$_{\mathcal{SB}}(p, j)$

Funkcija NASLEDNJA_POVEZAVA$_{\mathcal{SB}}(p, j)$ v vozlišču u (cilj paketa: v)

NASLEDNJA_POVEZAVA$_{\mathcal{SB}}(p, j)$:

```
1: if (j == 1) or (j == -1) then
2:     return ( sign(j) == sign(p.s))
3: else
4:     return ( sign(j) == sign(p.b))
```

Funkcija NASLEDNJA_POVEZAVA$_{\mathcal{SB}}(p, j)$ vrne rezultat na podlagi vrednosti paketu pripetih števil s in b (glej algoritem 4.1-II) in sicer takole: v primeru $j = 1$ (vprašanje se v tem primeru glasi: Ali na poti paketa p do cilja obstaja kratka povezava naprej?) funkcija vrne True tedaj in le tedaj, ko je $s > 0$; v primeru $j = -1$ funkcija vrne True tedaj in le tedaj, ko $s < 0$; podobno velja za $j = 2$ in $j = -2$ – v omenjenih primerih funkcija vrne rezultat na podlagi predznaka števila b.

4.1.2 Dinamičen algoritem

Za razliko od pravkar predstavljene statične izvedbe izračuna funkcije NASLEDNJA_POVEZAVA, pri kateri se je glavni del računanja opravil v času predprocesiranja, bomo v tem razdelku predstavili povsem dinamičen pristop – za izračun vrednosti funkcije NASLEDNJA_POVEZAVA$_{\mathcal{RSP}}$ na posameznem usmerjevalnem koraku potrebujemo samo podatek o trenutnem položaju paketa in o njegovem cilju. Dodatni čas, ki ga porabimo za omenjeno računanje, lahko opravičimo s kakovostjo rezultata. Pri statičnem algoritmu namreč na posameznem usmerjevalnem koraku izbiramo le med povezavami, ki pripadajo potem na začetku izbranega ekvivalenčnega razreda, dinamičen algoritem pa zazna tudi morebiten obstoj neekvivalentnih sprehodov in omogoča prehod med ekvivalenčnimi razredi. Moč množice povezav, ki so kandidati za nadaljevanje poti, je pri dinamičnem algoritmu v nekaterih primerih večja kot pri statičnem, kar omogoča večjo prilagodljivost in več možnosti za reševanje morebitnih težav (okvara omrežja, zasedenost povezave in podobno). Dinamičen algoritem, ki ga bomo predstavili v nadaljevanju, množico vseh kandidatov za nadaljevanje poti izračuna v času $\mathcal{O}(\log n)$.

Algoritem temelji na uporabi omejenih najkrajši poti in sicer uporablja naslednji preprost razmislek: če je za poljuben par števil i in j iz I dolžina omejene najkrajše poti $p_i^{(u,v)}$ večja od dolžine omejene najkrajše poti $p_j^{(u,v)}$, torej

$$d_i^{(u,v)} > d_j^{(u,v)},$$

potem na najkrajši poti med vozliščema u in v gotovo obstaja povezava tipa $\mathcal{E}_i$. Res, saj je najkrajša med vsemi potmi, ki ne uporabljajo povezave tipa $\mathcal{E}_i$ ($p_i^{(u,v)}$) daljša od neke

druge poti ($p_j^{(u,v)}$), torej ni najkrajša. Če naj paket ostane na najkrajši poti, *mora* narediti vsaj en korak po povezavi tipa $\mathcal{E}_i$.
V primeru pa, ko so vsa števila $d_i^{(u,v)}$, $i \in I$, enaka, algoritem določi množico kandidatov za nadaljevanje poti po naslednjem premisleku: najkrajša pot med u in v je gotovo enaka vsaj eni od omejenih najkrajših poti, saj sta vzdolž najkrajše poti kvečjemu dva različna tipa povezav; če so dolžine vseh omejenih najkrajših poti enake, so vse omejene najkrajše poti tudi najkrajše poti; vsaka povezava, ki v tem primeru sestavlja posamezno omejeno najkrajšo pot, je kandidat za naslednjo povezavo na najkrajši poti do cilja.

Algoritem 4.1-III NASLEDNJA_POVEZAVA$_{\mathcal{RSP}}(p, j)$

Računanje množice kandidatov $KND^{(u,v)}$ za potovanje paketa med vozliščema u in v ter pripadajoča funkcija NASLEDNJA_POVEZAVA$_{\mathcal{RSP}}(p, j)$

$KND^{(u,v)}$:

1: **for** $i \in I$ **do**
2: izračunaj $p_i^{(u,v)}$ in $d_i^{(u,v)}$;
3: $d^{(u,v)} := \min\{d_i^{(u,v)}; i \in I\}$; $D^{(u,v)} := \max\{d_i^{(u,v)}; i \in I\}$;
4: $C_1 := \{\mathcal{E}_i;\ i \in I; d_i^{(u,v)} > d^{(u,v)}\}$;
5: $C_2 := \{\mathcal{E}_i;\ \exists j \in I : d_j^{(u,v)} = d^{(u,v)}$ in $\mathcal{E}_i$ je povezava vzdolž poti $p_j^{(u,v)}\}$;
6: **return** $C_1 \cup C_2$.

NASLEDNJA_POVEZAVA$_{\mathcal{RSP}}(p, j)$:

1: **return** ($\mathcal{E}_j \in KND^{(u,v)}$);

V nadaljevanju bomo dokazali pravilnost celotnega dinamičnega postopka za izračun vrednosti funkcije NASLEDNJA_POVEZAVA$_{\mathcal{RSP}}$, ki ga prikazuje algoritem 4.1-III. Pri tem bomo uporabili naslednje oznake: s $P_i^{(u,v)}$ bomo označili množico vseh omejenih najkrajših poti tipa i med vozliščema u in v, z $d^{(u,v)}$ in $D^{(u,v)}$ pa dolžino najkrajše oziroma najdaljše (po posameznem tipu) omejene najkrajše poti (glej razdelek 3.12). Najkrajša pot med vozliščema u in v leži vsaj v eni od množic $P_i^{(u,v)}$, $i \in I$, v kar se prepričamo z naslednjo lemo.

Lema 4.1-I. *Vsaka najkrajša pot med vozliščema u in v dvokrožnega grafa je regularna in leži v množici $\bigcup_{i\in I} P_i^{(u,v)}$.*

Dokaz. Naj bo p najkrajša pot med vozliščema u in v. Ker $u \neq v$, pot p vsebuje vsaj eno povezavo, recimo povezavo tipa $\mathcal{E}_j$. Če bi p vsebovala tudi povezavo tipa $\mathcal{E}_{-j}$, p ne bi bila najkrajša, kar je v nasprotju s predpostavko. Pot p je torej regularna in leži v množici $P_{-j}^{(u,v)}$, zato $p \in \bigcup_{i\in I} P_i^{(u,v)}$. □

Delovanje algoritma 4.1-III temelji na trditvah 4.1-II in 4.1-IV, ki ju uporabimo pri konstrukciji množic kandidatov C_1 in C_2 za nadaljevanje poti v posledicah 4.1-III in 4.1-V.

Trditev 4.1-II. *Če je za poljubna $i, j \in I$ omejena najkrajša pot $p_i^{(u,v)}$ daljša od omejene najkrajše poti $p_j^{(u,v)}$ (t.j. $d_i^{(u,v)} > d_j^{(u,v)}$), potem vsaka najkrajša pot med vozliščema u in v vsebuje vsaj eno povezavo tipa $\mathcal{E}_i$.*

Dokaz. Recimo, da velja $d_i^{(u,v)} > d_j^{(u,v)}$ in da obstaja taka najkrajša pot p med vozliščema u in v, ki ne vsebuje povezave $\mathcal{E}_i$. Ker je $p \in P_i^{(u,v)}$, je dolžina poti p enaka $d_i^{(u,v)}$. Poleg tega je p najkrajša pot, torej je kvečjemu tako dolga kot $p_j^{(u,v)}$, zato velja $|p| = d_i^{(u,v)} \leq d_j^{(u,v)}$, kar pa je v nasprotju s predpostavko. $\rightarrow\!\!\!\!\!\!\times\!\!\!\!\!\!\leftarrow$ □

Posledica 4.1-III. *Kandidat za naslednjo povezavo na najkrajši poti med u in v je katerakoli povezava množice $C_1 := \{\mathcal{E}_i;\ i \in I; d_i^{(u,v)} > d^{(u,v)}\}$.*

Dokaz. Naj bo $\mathcal{E}_i$ element množice C_1 in naj za $j \in I$ velja $d_j^{(u,v)} = d^{(u,v)}$. Ker je $d_i^{(u,v)} > d^{(u,v)}$, torej $d_i^{(u,v)} > d_j^{(u,v)}$, je po trditvi 4.1-II na najkrajši poti med u in v vsaj ena povezava tipa $\mathcal{E}_i$. Ker vrstni red povezav vzdolž poti v krožnih grafih ni pomemben (glej trditev 3.5-I na strani 32), je $\mathcal{E}_i$ kandidat za naslednjo povezavo. □

Primer 4.1-II. Oglejmo si uporabo posledice 4.1-III na grafu $G(8;\ \pm 2, \pm 3)$. Omejene najkrajše poti med vozliščema 0 in 1 so $p_{-2}^{(0,1)} = (-1, 1),\ p_{-1}^{(0,1)} = (0, 3),\ p_1^{(0,1)} = (-1, 1)$ in $p_2^{(0,1)} = (2, -1)$. Ker v tem primeru velja $d^{(0,1)} = 2 < D^{(0,1)} = 3$ in $d_{-2}^{(0,1)} = d_1^{(0,1)} < d_{-1}^{(0,1)} = d_2^{(0,1)}$, nam posledica 4.1-III zagotavlja, da ob uporabi ene izmed povezav $\mathcal{E}_{-1}$ ali $\mathcal{E}_2$ ostanemo na najkrajši poti med 0 in 1.

Trditev 4.1-IV. *Dolžina najkrajše poti med vozliščema u in v je $d^{(u,v)}$.*

Dokaz. Najkrajša pot med u in v po lemi 4.1-I leži v eni od množic $P_i^{(u,v)}$, zato je njena dolžina enaka $\min_{i \in I} d_i^{(u,v)} = d^{(u,v)}$. □

Posledica 4.1-V. *Kandidat za naslednjo povezavo na najkrajši poti med u in v je vsaka povezava množice $C_2 := \{\mathcal{E}_i;\ \exists j \in I : d_j^{(u,v)} = d^{(u,v)}$ in $\mathcal{E}_i$ je povezava vzdolž poti $p_j^{(u,v)}\}$.*

Dokaz. Naj bo $l \in I$ tak, da je dolžina omejene najkrajše poti $d_l^{(u,v)}$ enaka $d^{(u,v)}$. Potem je po trditvi 4.1-IV omejena najkrajša pot $p_l^{(u,v)}$ tudi najkrajša pot med u in v. Množica C_2 torej vsebuje povezave, ki so sestavni del najkrajših poti med u in v, zato so vse povezave množice C_2 kandidati za naslednji korak. □

Primer 4.1-III. Oglejmo si dva primera uporabe posledice 4.1-V.
[1.] Vzemimo graf $G(8;\ \pm 2, \pm 3)$. Omejene najkrajše poti med vozliščema 0 in 4 so $p_{-2}^{(0,4)} = (2,0),\ p_{-1}^{(0,4)} = (2,0),\ p_{1}^{(0,4)} = (-2,0)$ in $p_{2}^{(0,4)} = (2,0)$. Ker je $d^{(0,4)} = D^{(0,4)} = 2$, je $d_i^{(0,4)} = d^{(0,4)}$ za vsak i. Posledica 4.1-V zato zagotavlja, da je prva povezava na najkrajši poti med 0 in 4 bodisi tipa $\mathcal{E}_1$ (ta povezava sestavlja poti $p_{-2}^{(0,4)}$, $p_{-1}^{(0,4)}$ in $p_{2}^{(0,4)}$) bodisi tipa $\mathcal{E}_{-1}$ (sestavni del poti $p_{1}^{(0,4)}$).

[2.] V grafu $G(10;\ \pm 3, \pm 4)$ so omejene najkrajše poti med vozliščema 0 in 5 $p_{-2}^{(0,5)} = (-1,2),\ p_{-1}^{(0,5)} = (1,-2),\ p_{1}^{(0,5)} = (-1,2)$ in $p_{2}^{(0,5)} = (1,-2)$. Ker je $d_i^{(0,5)} = d^{(0,5)}$ za vsak i in ker so v omejenih najkrajših poteh vključeni vsi tipi povezav ($\mathcal{E}_{-2}$ v poteh p_{-1} in p_2, $\mathcal{E}_{-1}$ v poteh p_{-2} in p_1, $\mathcal{E}_1$ v poteh p_{-1} in p_2 ter $\mathcal{E}_2$ v poteh p_{-2} in p_1), lahko za nadaljevanje poti uporabimo katerokoli povezavo.
Ta primer prikazuje moč posledice 4.1-V – če so namreč omejene najkrajše poti dovolj različne, lahko množica kandidatov vsebuje povezave dveh ali celo treh ekvivalenčnih razredov najkrajših poti (v našem primeru sta to razreda (1,-2) in (-1,2)).

Opomba. Pri izbiri omejene najkrajše poti imamo običajno na razpologo več možnosti. Tako bi lahko, na primer, v prvem primeru za $p_{-2}^{(0,4)}$ namesto (2,0) izbrali (-2,0). Katero pot izberemo, je odvisno od implementacije.

Množica kandidatov C_2 ni nikoli prazna, saj vsaj za eno omejeno najkrajšo pot velja, da je njena dolžina enaka $d^{(u,v)}$ (pravzaprav to velja vsaj za dve omejeni najkrajši poti – glej lemo 3.12-III). Množica kandidatov C_1 je prazna v primeru, ko je $d^{(u,v)} = D^{(u,v)}$, v nasprotnem primeru pa vsebuje en ali dva elementa. S sistematičnim pregledovanjem vseh dvokrožnih grafov s številom točk med 5 in 50 smo ugotovili, da ima množica kandidatov $KND^{(u,v)}$ v 36,5% moč 1, v 52,3% moč 2, v 8,3% moč 3, v preostalih 2,9% pa moč 4. Množica kandidatov za nadaljevanje je torej vsaj v 11,2% večja od množice kandidatov med katerimi izbira statičen algoritem, saj se ta omeji na en ekvivalenčni razred (torej na povezave največ dveh tipov).

Posledici 4.1-III in 4.1-V zagotavljata, da algoritem 4.1-III pravilno določi množico kandidatov za nadaljevanje poti $KND^{(u,v)}$. Ker se števili $d^{(u,v)}$ in $D^{(u,v)}$ (algoritem 4.1-III, vrstica 3) in množici C_1 ter C_2 (vrstica 4 in 5) izračunajo v konstantnem času (neodvisno od velikosti števila n), je časovna zahtevnost algoritma odvisna le od časa, ki je potreben za izračun omejenih najkrajših poti $p_i^{(u,v)}$ (vrstica 2). V razdelku 3.13 smo videli, da znamo te poti izračunati v logaritemskem času, torej je skupna časovna zahtevnost algoritma 4.1-III enaka $\mathcal{O}(\log n)$.

4.2 Splošni usmerjevalni problem

Pri splošnem usmerjevalnem problemu je skupaj v vseh vozliščih omrežja N paketov ($N \geq 1$). Vsak paket ima natančno določen cilj. Naloga usmerjevalnega algoritma za ta problem je, da v čim manjšem številu usmerjevalnih korakov vse pakete pripelje do njiho-

vega cilja. Pri tem mora upoštevati omejitve, ki jih narekujeta omrežje in izbrani usmerjevalni model. Za razliko od algoritmov za reševanje osnovnega problema mora algoritem za reševanje splošnega usmerjevalnega problema poleg funkcije, ki paketom zagotavlja, da ti ostanejo na najkrajši poti, predvideti tudi način za preprečevanje in reševanje posledic trkov.

4.2.1 Model usmerjevalnih algoritmov

Model usmerjevalnih algoritmov je ogrodje za zapis algoritmov za reševanje splošnega usmerjevalnega problema. Z uporabo izraza, izposojenega s področja objektnega programiranja, lahko rečemo, da je model usmerjevalnih algoritmov *abstrakten algoritem*. Povzema namreč zgradbo in delovanje algoritma, manjkajo pa mu nastavitve ključnih parametrov, ki opisujejo vedenje konkretnega algoritma. Algoritem opišemo tako, da k danemu modelu podamo nabor parametrov:

$$algoritem = model + parametri$$

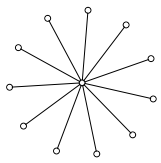
Model usmerjevalnih algoritmov omogoča:

a. **Enoten zapis usmerjevalnih algoritmov.** Algoritmi, zapisani na enoten način, so med seboj dobro primerljivi, poleg tega lahko natančno ugotovimo, katera lastnost posameznega algoritma vpliva na njegovo kakovost.

b. **Preprosto implementacijo izdelanih algoritmov v simulatorju.** Posamezen algoritem v simulatorju implementiramo tako, da podamo pripadajoči nabor parametrov modela. Izdelava novih algoritmov in njihova implementacija v simulatorju (in s tem hitra empirična ocena kakovosti) je s pomočjo modela izredno preprosta.

Pri razvoju usmerjevalnih algoritmov za omrežja s topologijo krožnih grafov s pridom izkoriščamo vozliščno simetrijo. Vsako vozlišče krožnega grafa lahko v procesu usmerjanja igra isto vlogo, oziroma, z drugimi besedami, v vseh vozliščih omrežja lahko teče isti usmerjevalni algoritem.

Primer 4.2-I. Drugače kot krožni grafi, omrežje *zvezda* predstavlja primer nesimetričnega omrežja. V centralnem vozlišču med procesom usmerjanja teče povsem drug algoritem kot v vseh drugih vozliščih.



Model usmerjevalnih algoritmov, predstavljen v tem razdelku, uporabljamo za zapis algoritmov, ki tečejo v posameznem vozlišču krožnega grafa. Ti algoritmi imajo nalogo, da v vsakem usmerjevalnem koraku čimveč paketov, ki so v njihovih čakalnih vrstah, po prostih povezavah pošljejo v sosednja vozlišča. Posamezen usmerjevalni algoritem je odgovoren le za pakete, ki so v njegovi čakalni vrsti, na potovanje drugih paketov ne more vplivati (razen posredno, ko lahko z določanjem visoke prioritete svojih paketov začasno

blokira uporabo katere od povezav). Skupna naloga vseh algoritmov v omrežju je, da paketom omogočijo čimprejšnji prihod na cilj; ko so tam vsi paketi, se postopek usmerjanja konča.

Zgradba usmerjevalnega algoritma
Usmerjevalni algoritem je sestavljen iz *predprocesiranja* in zaporedja *usmerjevalnih korakov* (glej sliko 4.2-I). V času predprocesiranja statični algoritmi izračunajo globalne

predprocesiranje;
while not *vsi_paketi_na_cilju* **do**
usmer. korak { *računanje_parametrov*; *izbira_paketov*; *pošiljanje_paketov*;
end;

Slika 4.2-I: Zgradba usmerjevalnega algoritma

usmerjevalne parametre – potrebovali jih bodo v nadaljnjem postopku usmerjanja. Dinamični algoritmi predprocesiranja ne uporabljajo. Usmerjevalni korak je sestavljen iz treh zaporednih, časovno usklajenih faz. V fazi *računanje_parametrov* dinamični algoritmi izračunajo usmerjevalne parametre, ki jih bodo potrebovali v *tem* usmerjevalnem koraku; statični algoritmi, ki so vse potrebne parametre (enkrat za vselej) izračunali že v času predprocesiranja, te faze ne uporabljajo.
Najpomembnejša faza celotnega usmerjanja je faza *izbira_paketov*, v kateri se sprejmejo vse pomembne odločitve: izberejo se paketi, ki bodo potovali v tem usmerjevalnem koraku, določi se povezave, po katerih bodo izbrani paketi potovali, razrešijo se morebitni trki na povezavah in podobno. V fazi *pošiljanje_paketov*, ki je enaka za vse algoritme, se dejansko izvrši prenos izbranih paketov. Do presenečenj v tej fazi ne more več priti, saj algoritem v fazi *izbira_paketov* za vsako povezavo določi kvečjemu po en paket (če je kandidatov za neko povezavo v danem usmerjevalnem koraku več, bo v fazi *pošiljanje_paketov* potoval tisti, ki ustreza optimizacijski strategiji danega algoritma – ta je določena z nekaterimi funkcijami faze *izbira_paketov*).

Čakalna vrsta
Vsako vozlišče omrežja je opremljeno s čakalno vrsto, kjer se zbirajo paketi, ki čakajo na nadaljevanje poti. Model predvideva čakalno vrsto, implementirano kot prioritetno vrsto, v kateri so paketi razporejeni glede na oddaljenost od cilja: paketi z največjo oddaljenostjo so na začetku, paketi z najmanjšo oddaljenostjo pa na koncu čakalne vrste. Paketi z isto oddaljenostjo od cilja so v vrsti razvrščeni po načelu FIFO (tisti, ki čakajo dalj časa, imajo prednost pred tistimi, ki so v vrsto prišli kasneje). Oddaljenost paketa od cilja določimo s pomočjo funkcije d_{cilj} – ker je podatek o oddaljenosti med dvema vozliščema globalna informacija o omrežju, funkcija d_{cilj}(paket) rezultat vrne v konstantnem času.

Algoritem 4.2-I PRVI_S_POVEZAVO(j)

Računanje vrednosti PRVI_S_POVEZAVO(j)

PRVI_S_POVEZAVO(j) :

```
1:  for i := 1 to stevilo_paketov_v_vrsti
2:      p := i-ti_paket_v_vrsti;
3:      if NASLEDNJA_POVEZAVA(p, j) then begin
4:          p odstrani_iz_vrste;
5:          return p
6:      end;
7:  end;
8:  // v vrsti ni primernega kandidata
9:  return null;
```

Čakalna vrsta je dodatno opremljena s funkcijo PRVI_S_POVEZAVO(j) (algoritem 4.2-I). Ta vrne prvi paket iz čakalne vrste (paketi se pregledujejo od začetka proti koncu vrste), ki lahko uporabi povezavo tipa j kot naslednjo povezavo na svoji poti do cilja; če v čakalni vrsti ni primernega paketa, funkcija vrne vrednost null. Paket, ki ga vrne funkcija PRVI_S_POVEZAVO, je izločen iz čakalne vrste, zato ga mora usmerjevalni algoritem, ki to funkcijo uporablja, bodisi poslati v sosednje vozlišče bodisi vstaviti nazaj v vrsto. Pri izbiri primernega kandidata za uporabo povezave tipa j funkcija PRVI_S_POVEZAVO uporablja funkcijo NASLEDNJA_POVEZAVA, ki je lastna posameznemu usmerjevalnemu algoritmu, torej eden od parametrov, s katerimi usmerjevalni algoritem v modelu opišemo. Funkcija NASLEDNJA_POVEZAVA(p, j) mora biti implementirana tako, da s True odgovori le, če je na poti paketa p gotovo še kakšna povezava tipa j in s False, če take povezave ni, oziroma če algoritem tega ne zna določiti.

Fazi *predprocesiranje* in *računanje_parametrov*
Model omogoča dva načina računanja usmerjevalnih parametrov – pred začetkom usmerjanja in na posameznem usmerjevalnem koraku. Statični algoritmi, ki vse parametre izračunajo pred začetkom usmerjanja, v modelu implementirajo samo funkcijo *predprocesiranje*, dinamični algoritmi, ki parametre računajo sproti, pa samo funkcijo *računanje_parametrov*. Algoritmi, ki nekaj dela opravijo pred začetkom usmerjanja, nekaj pa sproti, na posameznem usmerjevalnem koraku (to so t.i. *kombinirani* algoritmi), implementirajo obe funkciji.

Faza *izbira_paketov* – srce algoritma
Osnovna naloga faze *izbira_paketov* je izmed vseh paketov, ki v čakalni vrsti čakajo na nadaljevanje poti, izbrati tiste, ki bodo pot nadaljevali v tem usmerjevalnem koraku, in za-

nje rezervirati odgovarjajoče povezave. V ta namen algoritem po vrsti izbira tipe povezav. Za izbrani tip povezave določi dva paketa – najprimernejša kandidata v čakalni vrsti za potovanje po povezavi izbranega tipa v smeri naprej in nazaj – ter jima izračuna prioriteto. Ker *izbira_paketov* poteka časovno usklajeno v vseh vozliščih omrežja, se za vse povezave izbranega tipa hkrati določijo morebitni kandidati za potovanje v tem usmerjevalnem koraku. Pri tem se lahko zgodi, da se za katero od povezav potegujeta dva paketa (vsak z ene strani povezave). Usmerjevalni algoritem zato v nadaljevanju faze *izbira_paketov* primerja prioriteto "svojega" paketa s prioriteto paketa v sosednjem vozlišču. Paket, ki ima višjo prioriteto, je zmagovalec in bo v tem koraku potoval, poraženec, torej paket z nižjo prioriteto, pa bo moral počakati na naslednjo priložnost. Ta se bo morda ponudila (odvisno od narave algoritma) že pri naslednjem tipu povezav ali pa šele na naslednjem usmerjevalnem koraku.

Algoritem 4.2-II

Faza *izbira_paketov* v usmerjevalnem koraku τ

```
1:  V :={};
2:  for l := 1 to k do begin
3:      j := VRSTNI_RED(l);
4:      p→ := cakalnavrsta.PRVI_S_POVEZAVO(j);
5:      p← := cakalnavrsta.PRVI_S_POVEZAVO(−j);
6:      r^u_j := PRIORITETA_PAKETA(p→, j, τ);
7:      r^u_{−j} := PRIORITETA_PAKETA(p←, −j, τ);
8:      b^u_j := null; b^u_{−j} := null;
9:      if r^u_j > r^{u+h_j}_{−j} then b^u_j :=p→
10:         else if PAKETE_VRAČAM_SPROTI then VSTAVI_V_VRSTO({p→})
11:             else V := V∪{p→};
12:     if r^u_{−j} > r^{u−h_j}_j then b^u_{−j} := p_{−j}
13:         else if PAKETE_VRAČAM_SPROTI then VSTAVI_V_VRSTO({p←})
14:             else V := V∪{p←};
15: end;
16: VSTAVI_V_VRSTO(V);
```

Faza *izbira_paketov* konkretnega usmerjevalnega algoritma je natančno določena z naslednjimi parametri (glej algoritem 4.2-II).

- **Vrstni red izbire tipov povezav.** Ta parameter, ki je podan kot permutacija VRSTNI_RED množice tipov povezav I^+, določa vrstni red izbire tipov povezav na po-

sameznem usmerjevalnem koraku. Če tip j pride na vrsto prej kot tip k (to je VRSTNI_RED$^{-1}(j) <$ VRSTNI_RED$^{-1}(k)$), potem bo paket, ki je hkrati kandidat za potovanje po povezavi tipa j in po povezavi tipa k, najprej poskusil potovati po povezavi tipa j. Povezave, ki pridejo na vrsto prej, imajo torej določeno prednost pred tistimi, ki pridejo na vrsto kasneje.

- **Način izbire najprimernejših kandidatov za potovanje po izbrani povezavi.** Paketa $p_{\rightarrow}$ in $p_{\leftarrow}$, to je kandidata za potovanje po povezavi izbranega tipa v smeri naprej in nazaj, izberemo s pomočjo čakalni vrsti vgrajene funkcije PRVI_S_POVEZAVO, ki rezultat vrne na podlagi vrednosti funkcije NASLEDNJA_POVEZAVA.

- **Funkcija za določanje prioritete izbranim paketom.** Funkcija PRIORITETA_PAKETA, ki jo uporabljamo za določanje prioritete paketa, mora biti definirana tako, da za dva paketa, ki se vsak s svoje strani potegujeta za isto povezavo, nikoli ne bo vrnila iste vrednosti – če bi se to zgodilo, se namreč algoritem ne bi znal odločiti, kateremu paketu naj da prednost (mrtva zapora).

- **Način vračanja paketov v čakalno vrsto.** Glede na način vračanja paketov v čakalno vrsto razlikujemo dva tipa algoritmov: algoritme, ki pakete vračajo šele po koncu faze *izbira_paketov*, in algoritme, ki pakete vračajo sproti. Prvi paketom ne omogočajo, da bi v istem usmerjevalnem koraku kandidirali za več povezav – če izgubijo bitko za neko povezavo, morajo čakati na naslednji usmerjevalni korak tudi takrat, ko so najprimernejši kandidat za katero od preostalih povezav. Drugi algoritmi pa s sprotnim vračanjem paketom po porazu omogočijo kandidaturo že pri naslednjem tipu povezav. Tip algoritma je določen s parametrom PAKETE_VRAČAM_SPROTI – algoritmi prvega tipa imajo vrednost tega parametra nastavljeno na False, algoritmi drugega tipa pa na True.

Za boljše razumevanje delovanja si oglejmo še nekatere značilnosti faze *izbira_paketov* (glej algoritem 4.2-II). Množica V predstavlja množico poraženih paketov in jo uporabljamo samo takrat, ko algoritem paketov ne vrača sproti (vrstici 11 in 14); paketi, ki se po porazu dodajo množici V, se v čakalno vrsto vrnejo tik pred koncem faze *izbira_paketov* (vrstica 16). Funkcija VSTAVI_V_VRSTO vstavlja poražene pakete nazaj v čakalno vrsto. Pri tem upošteva prioriteto vrste (vrstni red paketov v vrsti je določen z oddaljenostjo paketov od cilja). Funkcijo VSTAVI_V_VRSTO uporabljajo algoritmi s sprotnim vračanjem (vrstici 10 in 13) in tudi tisti z zakasnenim (vrstica 16). Poleg množice V algoritem uporablja še naslednje lokalne spremenljivke: $p_{\rightarrow}$ in $p_{\leftarrow}$ ter tabelo izhodnih vmesnih pomnilnikov b. Prvi dve uporabljamo za začasno shranjevanje paketov, določenih za potovanje po izbranem tipu povezav, tabelo vmesnih pomnilnikov b pa za shranjevanje paketov, ki so uspešno prestali bitko za povezavo. Paketi počakajo v izhodnih vmesnih pomnilnikih do faze *pošiljanje_paketov*, v kateri nadaljujejo potovanje. Zelo pomembna za delovanje algoritma je tudi globalna dvodimenzionalna tabela r, ki jo uporabljamo za izmenjavo podatkov o prioritetah paketov. Vsako vozlišče piše le v svoj del tabele (vozlišču u so za pisanje dostopne le celice $r^u_j;\ j \in I$), bere pa lahko celotno tabelo. V vrsticah 6 in 7

algoritmi časovno usklajeno napolnijo vsak svoj del tabele, v vrsticah 9 in 12 pa na podlagi napisanih vrednosti določijo zmagovalne pakete. Do konfliktov pri pisanju v oziroma branju iz tabele r ne more priti, saj zaradi časovne usklajenosti v vsakem trenutku iz nje bodisi bere, bodisi vanjo piše kvečjemu en algoritem.

Faza *pošiljanje_paketov*
Zadnja faza usmerjevalnega koraka, to je faza *pošiljanje_paketov*, poskrbi, da se paketi, določeni za potovanje, premaknejo po izbranih povezavah v sosednja vozlišča. V tej fazi do trkov ne more priti, saj so se v fazi *izbira_paketov* morebitni trki že razrešili.

Algoritem 4.2-III

Faza *pošiljanje_paketov* v vozlišču u

```
1:  for j ∈ I do
2:      if b_j ≠ null then begin
3:          BO_PREMAKNJEN(b_j, u, j);
4:          POSLJI(b_j, u, u + sign(j) * h_|j|);
5:      end;
```

Za vsak pripravljen paket, ki čaka v odgovarjajočem izhodnem vmesnem pomnilniku, algoritem v fazi *pošiljanje_paketov* izvede dve funkciji (glej algoritem 4.2-III), in sicer:

- BO_PREMAKNJEN(p, u, j). Funkcija BO_PREMAKNJEN, ki je lastna posameznemu usmerjevalnemu algoritmu, omogoča spremembo parametrov, ki so vezani na položaj paketa. Pri statičnih algoritmih je ta funkcija implementirana tako, da spremeni parametre, ki so pripeti k paketu, pri dinamičnih algoritmih pa ne naredi ničesar. Vhodni parametri funkciji BO_PREMAKNJEN so paket p, vozlišče u, v katerem je p pred pošiljanjem, in tip povezave j, po kateri bo p poslan.

- POSLJI(p, u, v). Funkcija POSLJI, ki je skupna vsem algoritmom, poskrbi, da se paket zares premakne iz vozlišča u v sosednje vozlišče v. Vhodni parametri so paket p, vozlišče u, v katerem je p pred pošiljanjem, in vozlišče v, v katerega bo paket poslan.

Zapis usmerjevalnega algoritma v modelu
Algoritem v modelu usmerjevalnih algoritmov zapišemo tako, da podamo odgovarjajočo funkcijo NASLEDNJA_POVEZAVA ter vrednosti parametrov, ki smo jih predstavili v tem razdelku. Tabela 4.2-I povzema omenjene parametre skupaj z njihovimi privzetimi vrednostmi. V tabeli smo "procedura brez učinka" označili proceduro, v kateri se nič ne zgodi (begin-end procedura), z "identiteta" pa funkcijo Id, za katero velja $Id(i) = i,\ \forall i \in I^+$).

parameter	privzeta vrednost
predprocesiranje	procedura brez učinka
računanje_parametrov	procedura brez učinka
VRSTNI_RED	identiteta
PRIORITETA_PAKETA	ni privzete vrednosti
PAKETE_VRAČAM_SPROTI	True
BO_PREMAKNJEN	procedura brez učinka

Tabela 4.2-I: Parametri modela in njihove privzete vrednosti

4.2.2 Statični usmerjevalni algoritmi $\mathcal{S}_0$, $\mathcal{S}_1$, $\mathcal{S}_2$, $\mathcal{S}_3$ in $\mathcal{S}_4$

V [16] smo predstavili razvoj petih statičnih usmerjevalnih algoritmov za dvokrožne grafe. Začeli smo z znanim usmerjevalnim algoritmom $\mathcal{S}_0$[55], nato smo postopoma spreminjali njegove lastnosti in sicer tako, da se je njegova kakovost pri vsaki spremembi v povprečju (glede na rezultate simulacij) izboljšala. Zadnji algoritem v tej verigi razvoja, to je algoritem $\mathcal{S}_4$, je, kot so pokazali rezultati obširnih simulacij, skoraj za 100% izboljšal kakovost začetnega algoritma $\mathcal{S}_0$.
Opomba. V [16] smo za poimenovanje statičnih algoritmov uporabili oznake `RouteOffDDA`, `RouteOffDDB`, `RouteOffDDC`, `RouteOffDDD` in `RouteOffDDE`, v pričujočem delu pa smo te oznake opustili, saj namesto njih uporabljamo (po vrsti) oznake $\mathcal{S}_0$, $\mathcal{S}_1$, $\mathcal{S}_2$, $\mathcal{S}_3$ in $\mathcal{S}_4$. Krajše poimenovanje statičnih algoritmov smo uvedli zaradi skladnosti z oznakami drugih algoritmov in oznakami, ki smo jih uporabili v [26].

V tem razdelku bomo na kratko predstavili omenjenih pet algoritmov in njihove zapise v modelu usmerjevalnih algoritmov. Videli bomo, da se algoritmi $\mathcal{S}_0$, $\mathcal{S}_1$, $\mathcal{S}_2$, $\mathcal{S}_3$ in $\mathcal{S}_4$ ujemajo v fazi *predprocesiranje*, permutaciji VRSTNI_RED ter v funkciji BO_PREMAKNJEN, razlikujejo pa se po načinu reševanja trkov in po strategiji določanja "pomembnosti" paketov (torej v parametru PAKETE_VRAČAM_SPROTI ter funkcijah NASLEDNJA_POVEZAVA in PRIORITETA_PAKETA).

Algoritmi $\mathcal{S}_0$, $\mathcal{S}_1$, $\mathcal{S}_2$, $\mathcal{S}_3$ in $\mathcal{S}_4$ so bili izdelani za reševanje permutacijskega usmerjevalnega problema, zato jih bomo v taki obliki predstavili tudi v tem razdelku. Predpostavili bomo, da je pred začetkom usmerjanja v vsakem vozlišču omrežja natanko en paket z enolično določenim ciljem (paket v vozlišču u je namenjen v vozlišče $\pi(u)$, pri čemer je π permutacija množice vozlišč omrežja, ki dani problem opisuje).

Predprocesiranje
Naši statični usmerjevalni algoritmi v času predprocesiranja določijo ekvivalenčni razred poti, po kateri bo posamezen paket potoval (glej razdelek 3.5). Algoritem, ki se izvaja v vozlišču u, izračuna par (s, b), ki predstavlja število kratkih in dolgih povezav vzdolž

najkrajše poti med vozliščema u in $\pi(u)$, ter par (s, b) kot dodatno informacijo pripne paketu. Spomnimo se, da sta števili s in b lahko tudi negativni (pozitivna vrednost predstavlja povezave v smeri "naprej", negativna pa v smeri "nazaj"). Dodatna informacija, ki bo do konca usmerjanja potovala skupaj s paketom, se med potjo primerno spreminja. Če se, na primer, paket v nekem usmerjevalnem koraku v sosednje vozlišče premakne po kratki povezavi naprej (nazaj), se zmanjša (zveča) vrednost števila s; podobno velja za število b in dolge povezave. Paket pride na cilj, ko obe števili s in b dosežeta vrednost 0. Kot smo videli v razdelku 3.8, znamo števili s in b izračunati v logaritmičnem času, zato je časovna zahtevnost predprocesiranja enaka $\mathcal{O}(\log n)$.

Algoritem $\mathcal{S}_0$
Delovanje algoritma $\mathcal{S}_0$ temelji na naslednji strategiji: paket lahko potuje po kratki povezavi šele tedaj, ko na preostanku poti do cilja ni več dolgih povezav. Ker paketi najprej naredijo dolge korake, ta način usmerjanja imenujemo *usmerjanje z dolgimi koraki* (*angl. big-foot routing*). Strategija dolgih korakov se izkaže za izredno učinkovito, če jo uporabimo na hkratno dvosmernem modelu omrežja [41] – zaradi uravnoteženega priliva in odliva paketov je v posameznem usmerjevalnem koraku v vsakem vozlišču največ en paket, ki je kandidat za potovanje po dolgi povezavi, zato do trkov na dolgih povezavah ne more priti; trke na kratkih pa lahko učinkovito rešujemo po kriteriju oddaljenosti od cilja (ker je pri paketih, ki smejo potovati po kratki povezavi, $b = 0$, je oddaljenost od cilja enaka $|s|$). To lastnost strategije dolgih korakov v neusmerjenem modelu ohranimo s simulacijo hkratno dvosmernega modela in sicer tako, da usmerjevalne korake razdelimo, glede na njihovo zaporedno številko, na sode in lihe ter uvedemo omejevanje gibanja paketov: v sodih usmerjevalnih korakih prepovemo potovanja po povezavah v smeri nazaj, v lihih pa po povezavah v smeri naprej [55]. Tako pridemo do algoritma $\mathcal{S}_0$, pri katerem do trkov na dolgih povezavah sicer res ne more priti, je pa časovno precej potraten. V najslabšem primeru namreč algoritem naredi dvakrat več usmerjevalnih korakov, kot bi jih naredil teoretično optimalen algoritem.

> Primer 4.2-II. Recimo, da imamo v omrežju le en paket s pripetim $(s, b) = (0, 2)$. Pot omenjenega paketa je torej sestavljena iz dveh dolgih povezav v smeri naprej. Optimalen algoritem bi paket pripeljal do cilja v dveh korakih, algoritem $\mathcal{S}_0$ pa potrebuje štiri – ker so v lihih usmerjevalnih korakih povezave v smeri naprej blokirane, bo paket potoval le v sodih korakih, torej v korakih številka 2 in 4 (v korakih številka 1 in 3 pa se ne bo zgodilo nič).

S pomočjo obširnih simulacij smo ugotovili, da algoritem $\mathcal{S}_0$ pri reševanju permutacijskega usmerjevalnega problema tudi v povprečnem primeru naredi skoraj dvakrat več usmerjevalnih korakov kot teoretično optimalen algoritem (glej rezultate simulacij, tabela 4.2-IV).

Algoritem $\mathcal{S}_1$
Slaba lastnost algoritma $\mathcal{S}_0$ je torej, da morajo paketi, ki se potegujejo za neko povezavo,

čakati na pravilen (sod ali lih) usmerjevalni korak tudi takrat, ko v nasprotni smeri ni tekmeca za izbrano povezavo. To slabost odpravlja algoritem $\mathcal{S}_1$, ki kriterij sodo/liho uporabi le v primeru trkov (to je, ko se za neko povezavo potegujeta dva kandidata); kadar pa je kandidat en sam, omejitev na povezavi ne upošteva. Tako odpravi nepotrebno čakanje (usmerjevalni problem s primera 4.2-II, na primer, algoritem $\mathcal{S}_1$ reši v dveh korakih), zagotovila, da do trkov na dolgih povezavah ne bo prišlo, pa žal ni več. Podobno kot $\mathcal{S}_0$ se vsi trki v vozliščih (to je, kadar se v istem vozlišču več kot en paket poteguje za isto povezavo) rešujejo po kriteriju oddaljenosti, to je glede na vrednost vsote $|s|+|b|$. Čeprav je na prvi pogled omenjena sprememba majhna, predstavlja veliko izboljšavo v kakovosti algoritma (glej rezultate simulacije, tabela 4.2-IV). Za implementacijo omenjene spremembe je treba v čakalno vrsto vgraditi bolj zapleten postopek izbiranja paketov. Medtem ko je bil pri algoritmu $\mathcal{S}_0$ zaradi uravnoteženega pretoka paketov za potovanje po dolgi povezavi na razpolago kvečjemu en kandidat (torej do trkov v vozlišču ni moglo priti), je lahko pri algoritmu $\mathcal{S}_1$ takih kandidatov več. Čakalna vrsta algoritma $\mathcal{S}_1$ med temi paketi izbira na podlagi oddaljenosti od cilja, oziroma pri paketih z enako oddaljenostjo na podlagi položaja v vrsti (FIFO).

algoritem	osnovna strategija	trki v vozliščih	trki na povezaveh	pre-usm.	pot paketov
$\mathcal{S}_0$	kratko povezavo uporabi šele, ko zmanjka dolgih	h_1: $\max\|s\|$ h_2: ni trkov	sodo/liho (dosledno)	ne	naprej dolge povezave (big-foot dosledno)
$\mathcal{S}_1$	odpravi nepotrebno sodo/liho čakanje	$\max(\|s\|+\|b\|)$ ali FIFO	sodo/liho (po potrebi)	$-ii-$	$-ii-$
$\mathcal{S}_2$	dovoli uporabo kratkih povezav tudi, če dolge so	$-ii-$	$-ii-$	$-ii-$	pot v ekvivalenčnem razredu
$\mathcal{S}_3$	uporaba kriterija oddaljenosti za reševanje trkov na povezavah	$-ii-$	največja razdalja ali sodo/liho	$-ii-$	$-ii-$
$\mathcal{S}_4$	omogoči preusmeritev paketov	$-ii-$	$-ii-$	da	$-ii-$

Tabela 4.2-II: Osnovne značilnosti algoritmov $\mathcal{S}_0$, $\mathcal{S}_1$, $\mathcal{S}_2$, $\mathcal{S}_3$, and $\mathcal{S}_4$

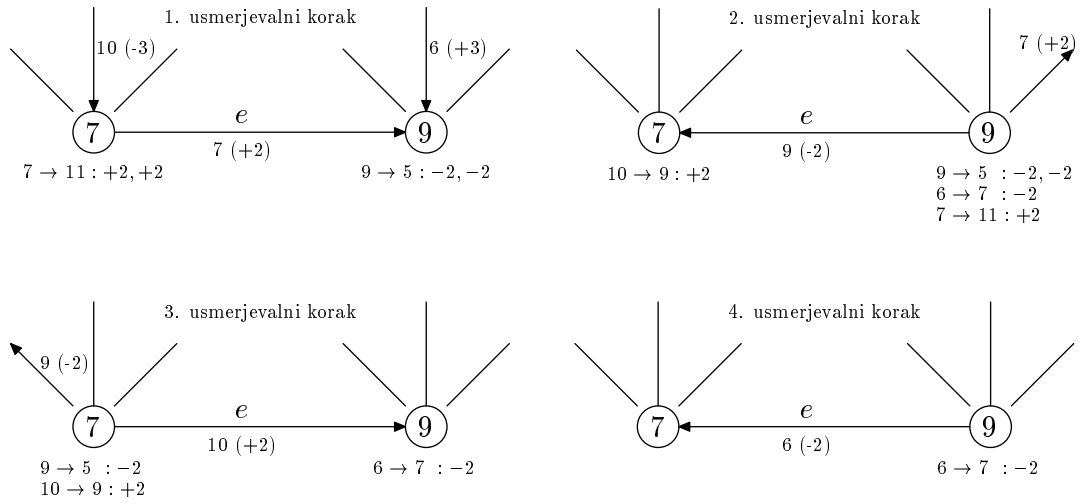
Algoritem $\mathcal{S}_2$
Uporaba strategije dolgih korakov, ki je pri algoritmu $\mathcal{S}_0$ posredno preprečevala trke na dolgih povezavah, se pri algoritmu $\mathcal{S}_1$ ne zdi več smiselna, saj ta, zaradi nedosledne uporabe sodih in lihih usmerjevalnih korakov, ponovno omogoči možnost omenjenih trkov.

Algoritem $\mathcal{S}_2$ se od algoritma $\mathcal{S}_1$ razlikuje le po tem, da po kratki povezavi dovoli potovati tudi paketom, ki še niso porabili vseh dolgih povezav. Čeprav omenjena sprememba prepreči nekatera nepotrebna čakanja, pa zaradi nje pride do nekaterih trkov, ki se prej ne bi zgodili. V povprečju je, kot pokažejo rezultati simulacij, algoritem $\mathcal{S}_2$ le za odtenek boljši od algoritma $\mathcal{S}_1$.

Algoritem $\mathcal{S}_3$
Pri razvoju algoritma $\mathcal{S}_3$ nas je vodila misel, da algoritem, pri katerem reševanje trkov na povezavah temelji le na kriteriju sodo/liho, ne more biti optimalen. V algoritem $\mathcal{S}_3$ smo torej vgradili strategijo, s pomočjo katere ta trke rešuje glede na oddaljenost paketov od cilja – v bitki za povezavo zmaga tisti paket, ki ima pred sabo daljšo pot; če imata oba paketa, ki se potegujeta za povezavo, pred sabo isto dolgo pot, zmaga tisti, ki glede na kriterij sodo/liho potuje v pravi smeri. Omenjena sprememba bistveno popravi kakovost algoritma (glej tabelo 4.2-IV).

Algoritem $\mathcal{S}_4$
Zadnja sprememba, ki smo jo naredili v razvoju algoritmov, je povezana s sposobnostjo večkratne uporabe istega paketa v nekem usmerjevalnem koraku. Vsi algoritmi doslej so namreč paketu, ki je izgubil bitko na povezavi, nadaljevanje potovanja omogočili šele v naslednjem usmerjevalnem koraku, pa čeprav bi bil ta paket morda dober kandidat za katero od drugih povezav. Algoritem $\mathcal{S}_4$ pa omogoča tako imenovano *preusmeritev* – paket, ki je izgubil bitko, se takoj vrne v čakalno vrsto in tako postane enakovreden tekmec za potovanje po drugih povezavah.

Čeprav je algoritem $\mathcal{S}_4$ bistvena izboljšava algoritma $\mathcal{S}_0$, $\mathcal{S}_4$ ni optimalen. Poznamo celo primere, pri katerih $\mathcal{D}$ opravi dvakrat več usmerjevalnih korakov kot bi bilo treba – časovna zahtevnost v najslabšem primeru se torej z razvojem algoritmov ni popravila. Naj omenimo še zanimivost, da smo s pomočjo simulatorja našli nekaj teh najslabših primerov (glej primer 4.2-III), in še to vse le takrat, ko je bila dolžina najdaljše poti do cilja enaka 2.



Slika 4.2-II: Primer usmerjanja z algoritmom $\mathcal{S}_4$ v omrežju $G(12;\ \pm 2, \pm 3)$

Primer 4.2-III. Oglejmo si reševanja permutacijskega usmerjevalnega problema, pri katerem algoritem $\mathcal{S}_4$ opravi dvakrat več usmerjevalnih korakov, kot je največja oddaljenost paketov od cilja. Vzemimo graf $G(12;\ \pm 2, \pm 3)$ in permutacijo $\pi = (0\ 3\ 8\ 1\ 10\ 9\ 5\ 6\ 7\ 11)(2\ 4)$. Pri taki razporeditvi paketov je največja oddaljenost od cilja enaka 2. Oglejmo si dogajanje v vozliščih 7 in 9 (slika 4.2-II). Preko teh dveh vozlišč so med usmerjanjem potovali paketi 6, 7, 9 in 10. Paket številka 6 je moral za dosego cilja opraviti dolg korak naprej in kratkega nazaj, 6: +3, -2. Podobno, 7: +2,+2, 9: -2,-2 in 10: -3,+2. Že v prvem usmerjevalnem koraku pride do bitke za povezavo e – potrebujeta jo namreč paketa 7 in 9; zmaga paket 7, ker potuje v pravi smeri (kriterij sodo/liho). Poleg paketa 7 v tem koraku potujeta tudi paketa 10 (10-3=7) in 6 (6+3=10). V drugem koraku se za povezavo e potegujejo paketi 10, 9 in 6 – zmaga paket številka 9, ker ima do cilja še dva koraka, tekmeca pa le po enega. V tretjem koraku se za povezavo e potegujeta paketa 9 in 6 – zmagovalca spet določi kriterij sodo/liho, zato potuje paket številka 10. Paket številka 6 je tako svoj drugi korak uspel opraviti šele v četrtem usmerjevalnem koraku.

Zapis algoritmov $\mathcal{S}_0$, $\mathcal{S}_1$, $\mathcal{S}_2$, $\mathcal{S}_3$ in $\mathcal{S}_4$ v modelu

Algoritmi $\mathcal{S}_0$, $\mathcal{S}_1$, $\mathcal{S}_2$, $\mathcal{S}_3$ in $\mathcal{S}_4$ se ujemajo v fazah *predprocesiranje*, *računanje_parametrov*, v permutaciji VRSTNI_RED in v metodi BO_PREMAKNJEN, razlikujejo pa se v implementaciji funkcij NASLEDNJA_POVEZAVA, PRIORITETA_PAKETA in vrednosti parametra PAKETE_VRAČAM_SPROTI, kot sledi.

Faza *predprocesiranje*. Za posamezen paket v tej fazi izračunamo par (s, b), ki predstavlja ekvivalenčni razred najkrajše poti med izvorom in ponorom, in ga pripnemo paketu. Za računanje uporabimo algoritem 3.11-III, predstavljen na strani 69.

Faza *računanje_parametrov*. Ker so algoritmi $\mathcal{S}_0$, $\mathcal{S}_1$, $\mathcal{S}_2$, $\mathcal{S}_3$ in $\mathcal{S}_4$ statični, faze *računanje_parametrov* ne uporabljajo.

Permutacija VRSTNI_RED. Permutacija množice tipov povezav, to je funkcija VRSTNI_RED, je v vseh petih algoritmih podana z

$$\text{VRSTNI_RED} = \begin{pmatrix} 1 & 2 \\ 2 & 1 \end{pmatrix}.$$

V prvi iteraciji algoritma 4.2-II torej določimo kandidate za potovanje po dolgih, v drugi pa kandidate za potovanje po kratkih povezavah. Čeprav smo že pri algoritmu $\mathcal{S}_2$ odpravili pogoj, da paketi potujejo po kratkih povezavah šele, ko zmanjka dolgih, imajo, zaradi implementacije funkcije VRSTNI_RED, pri vseh algoritmih dolge povezave še vedno majhno prednost pred kratkimi. Če je namreč v nekem usmerjevalnem koraku paket enako dober kandidat za potovanje po dolgi in kratki povezavi, potem bo v tem koraku potoval po dolgi povezavi. Tak način izbire funkcije VRSTNI_RED delno upravičimo z naslednjim argumentom: naši algoritmi težijo k lokalizaciji problema – s prvimi koraki paketi po dolgih povezavah prodrejo v bližnjo okolico cilja, v nadaljnjih korakih pa po kratkih povezavah v točno določeno vozlišče.

Funkcija NASLEDNJA_POVEZAVA(p, j). Algoritma $\mathcal{S}_0$ in $\mathcal{S}_1$, ki dovoljujeta uporabo kratkih povezav samo takrat, ko na poti do cilja ni več dolgih, uporabljata strategijo za izračun funkcije NASLEDNJA_POVEZAVA$_{\mathcal{BF}}$, kot je prikazana v algoritmu 4.2-IV.

Algoritem 4.2-IV NASLEDNJA_POVEZAVA$_{\mathcal{BF}}(p, j)$

Računanje vrednosti NASLEDNJA_POVEZAVA$_{\mathcal{BF}}(p, j)$ za $\mathcal{S}_0$ in $\mathcal{S}_1$

NASLEDNJA_POVEZAVA$_{\mathcal{BF}}(p, j)$:

```
1:  if (j == 1) or (j == -1) then
2:      return (p.b==0) and ( sign(j) ==  sign(p.s))
3:  else
4:      return ( sign(j) ==  sign(p.b)).
```

Dolge povezave ($j = 2$ ali $j = -2$) so torej vzdolž poti, če ima parameter b pravilen predznak (glede na j), kratke povezave ($j = 1$ ali $j = -1$) pa, če je predznak s pravilen in je poleg tega $b = 0$ (ni več dolgih).
Implementacija funkcije NASLEDNJA_POVEZAVA se pri algoritmih $\mathcal{S}_2$, $\mathcal{S}_3$ in $\mathcal{S}_4$, ki ne prepovedujejo uporabe kratkih povezav, razlikuje le v delu druge vrstice algoritma 4.2-IV, ki zahteva $b == 0$ – opustitev tega pogoja pripelje do algoritma NASLEDNJA_POVEZAVA$_{\mathcal{SB}}$ (glej stran 102), ki ga ti algoritmi uporabljajo.

Funkcija PRIORITETA_PAKETA. Računanje prioritete paketom se je med razvojem algoritmov dvakrat spremenilo. Algoritem $\mathcal{S}_0$ uporablja preprosto funkcijo, ki prioriteto

določi samo po kriteriju sodo/liho – za pakete, ki glede na sodost/lihost usmerjevalnega koraka želijo potovati v pravi smeri, funkcija vrne vrednost 1, za druge pa 0. Tako so paketi, ki želijo potovati v napačni smeri, izenačeni z "neobstoječimi" paketi – če namreč v vozlišču ni kandidata za potovanje v določeni smeri, se prioriteta po privzetku nastavi na vrednost 0. V zapisu algoritma spodaj smo s k označili zaporedno številko usmerjevalnega koraka.

A1: PRIORITETA_PAKETA(p,j,k) :=
A2: ($p == null$) ? 0 : ((parity(k) == sign(j)) ? 1 : 0);

Že pri algoritmu $\mathcal{S}_1$ smo, da bi odpravili odvečno čakanje, spremenili način računanja prioritete in sicer tako, da algoritem jasno loči med pravimi in neobstoječimi paketi. Funkcija PRIORITETA_PAKETA torej za algoritem $\mathcal{S}_1$ vrne vrednost 0, če ni kandidata za potovanje po izbrani povezavi; 1, če kandidat je, vendar želi potovati v napačni smeri in 2, če želi kandidat potovati v pravi smeri. Prednost pri potovanju ima tako kandidat, ki potuje v pravi smeri, če pa takega ni, lahko potujejo tudi drugi.

B1: PRIORITETA_PAKETA(p,j,k) :=
B2: ($p == null$) ? 0 : ((parity(k) == sign(j)) ? 2 : 1);

Enako funkcijo za določanje prioritete uporablja tudi algoritem $\mathcal{S}_2$; spremeni se šele pri algoritmu $\mathcal{S}_3$ (kriterij sodo/liho zamenjamo s kriterijem oddaljenosti od cija). Namesto preprostih vrednosti 0,1 in 2 rezultat funkcije PRIORITETA_PAKETA pri $\mathcal{S}_3$ in $\mathcal{S}_4$ nosi informacijo o paketovi oddaljenosti od cilja. Vrednost prioritete algoritem izračuna tako, da paketovo oddaljenost pomnoži z dva in ji prišteje 1, če želi paket glede na kriterij sodo/liho potovati v pravi smeri. Tako dosežemo, da v bitki za povezavo zmaga paket z večjo oddaljenostjo od cilja, če pa se za povezavo potegujeta enako oddaljena paketa, zmaga tisti, ki potuje v "pravi" smeri.

D1: PRIORITETA_PAKETA(p,j,k) :=
D2: ($p == null$) ? 0 : 2 * d_{cilj}+ ((parity(k) == sign(j)) ? 1 : 0);

V zgornji formuli smo z d_{cilj} označili oddaljenost paketa od cilja, to je razdaljo med trenutnim položajem paketa in njegovim ciljem; podatek o tej oddaljenosti je del globalne informacije o omrežju.

PAKETE_VRAČAM_SPROTI Vsi algoritmi, razen $\mathcal{S}_4$, poražene pakete v čakalno vrsto vračajo šele po koncu faze *izbira_paketov* – pri njih je torej vrednost parametra PAKETE_VRAČAM_SPROTI enaka False, pri algoritmu $\mathcal{S}_4$ pa True. Nastavitev vrednosti parametra PAKETE_VRAČAM_SPROTI je edina razlika med zadnjima algoritmoma $\mathcal{S}_3$ in $\mathcal{S}_4$.

BO_PREMAKNJEN Algoritmi $\mathcal{S}_0$, $\mathcal{S}_1$, $\mathcal{S}_2$, $\mathcal{S}_3$ in $\mathcal{S}_4$ funkcijo BO_PREMAKNJEN uporabljajo za korekcijo vrednosti parametrov s in b. Če bo, na primer, paket premaknjen v smeri povezave tipa 1, funkcija BO_PREMAKNJEN zmanjša vrednost parametra s; podobno velja za druge tipe povezav, kot vidimo iz spodnje kode.

```
BO_PREMAKNJEN(p, u, j):
  case j of
    -2: p.b := p.b + 1;
    -1: p.s := p.s + 1;
     1: p.s := p.s − 1;
     2: p.b := p.b − 1
  end;
```

4.2.3 Dinamični usmerjevalni algoritem $\mathcal{D}$

Z razvojem dinamičnega algoritma smo želeli ustvariti algoritem, ki za določanje nadaljevanja poti posameznega paketa ne potrebuje nobene vnaprej pripravljene informacije – niti v obliki usmerjevalnih tabel niti v obliki informacij, pridobljenih v času predprocesiranja. Rezultat razvoja je algoritem $\mathcal{D}$, ki na posameznem usmerjevalnem koraku za določitev naslednje povezave na najkrajši poti potrebuje le podatek o trenutnem položaju in cilju danega paketa. Tak algoritem se dobro znajde tudi v situaciji, ko se med samim potekom usmerjanja v omrežju pojavijo novi paketi; statičen algoritem tega ne zmore, saj novi paketi niso prestali obvezne faze predprocesiranja. Dodatna prednost algoritma $\mathcal{D}$ pred statičnimi algoritmi $\mathcal{S}_0$, $\mathcal{S}_1$, $\mathcal{S}_2$, $\mathcal{S}_3$ in $\mathcal{S}_4$ izvira iz načina določanja kandidatov za naslednjo povezavo na najkrajši poti. Medtem, ko pri omenjenih statičnih algoritmih kandidate za nadaljevanje poti izbiramo med povezavami na začetku določenega ekvivalenčnega razreda, imamo pri dinamičnem algoritmu možnost, da med samim usmerjanjem ekvivalenčni razred zamenjamo. To nam zagotavlja večje število kandidatov za nadaljevanje poti in tako več možnosti za preprečevanje trkov in reševanje posledic morebitnega izpada katere od povezav.

Osnovna misel našega dinamičnega pristopa je: nadaljevanje poti za dani paket določim sproti – na trenutnem usmerjevalnem koraku – s pomočjo metode omejenih najkrajših poti. Rezultat te metode, ki smo jo podrobneje predstavili v razdelku 4.1.2, je množica povezav, ki so kandidati za naslednjo povezavo vzdolž najkrajše poti. Izmed teh povezav algoritem izbere eno in sicer tako, da bo skupno število trkov čim manjše. Pri tem uporablja isti pristop kot statični algoritem $\mathcal{S}_4$. Natančneje bomo algoritem $\mathcal{D}$ podali z opisom načina za preprečevanje in reševanje posledic trkov ter načina za določanje najkrajše poti, kot sledi.

Način preprečevanja in reševanja posledic trkov
Ker smo na podlagi obširnih simulacij algoritmov $\mathcal{S}_0$, $\mathcal{S}_1$, $\mathcal{S}_2$, $\mathcal{S}_3$ in $\mathcal{S}_4$ (glej razdelek 4.2.4) videli, da je najučinkovitejši sistem za reševanje in preprečevanje posledic trkov tisti, ki

ga uporablja algoritem $\mathcal{S}_4$, smo se odločili, da se bo algoritem $\mathcal{D}$ s tem problemom spopadel na enak način. Vrstni red izbire tipov povezav je zato pri algoritmu $\mathcal{D}$, enako kot pri algoritmu $\mathcal{S}_4$, določen s permutacijo $(2\ 1)$, paketi se v vrsto vračajo sproti, prioriteta paketa pa je določena glede na njegovo oddaljenost od cilja in glede na smer, v kateri želi potovati (glej vrstico D2 funkcije PRIORITETA_PAKETA na strani 118). Algoritma $\mathcal{S}_4$ in $\mathcal{D}$ se torej razlikujeta le po načinu za določanje najkrajše poti — primerjava delovanja obeh algoritmov zato pokaže, kako ta razlika vpliva na kakovost usmerjanja.

Način za določanje najkrajše poti
Algoritem $\mathcal{D}$ s pomočjo metode omejenih najkrajših poti, ki smo jo predstavili v razdelku 4.1.2, posameznemu paketu zagotavlja, da ta ostane na najkrajši poti. Faze *predprocesiranje* ne uporablja, saj vse podatke pripravi v fazi *računanje_parametrov* in sicer tako, da za vsak paket v čakalni vrsti danega vozlišča u s ciljem v določi množico kandidatov za naslednjo povezavo $KND^{(u,v)}$(algoritem 4.1-III, vrstice 1–7). V fazi *izbira_paketov* nato izbiro paketov v čakalni vrsti opravi s pomočjo funkcije NASLEDNJA_POVEZAVA$_{\mathcal{RSP}}$ (algoritem 4.1-III, vrstica 8).

Primerjava algoritmov $\mathcal{S}_4$ in $\mathcal{D}$
Algoritma $\mathcal{S}_4$ in $\mathcal{D}$ se ujemata v načinu za preprečevanje in reševanje posledic trkov, razlikujeta pa v načinu za določanje nadaljevanja poti (predvsem v implementaciji funkcije NASLEDNJA_POVEZAVA). Tabela 4.2-III prikazuje nastavitve parametrov modela za oba algoritma, v razdelku 4.2.4 pa so primerjave učinkovitosti obeh algoritmov pri izvajanju v simulatorju.

parameter	algoritem $\mathcal{S}_4$	algoritem $\mathcal{D}$
VRSTNI_RED	permutacija $(2\ 1)$	permutacija $(2\ 1)$
PRIORITETA_PAKETA	D2 (glej str. 118)	D2 (glej str. 118)
PAKETE_VRAČAM_SPROTI	True	True
predprocesiranje	izračuna par (s, b) in ga pripne paketu	(ne uporablja)
računanje_parametrov	(ne uporablja)	za vsak paket v vrsti izračuna množico $KND^{(u,v)}$
BO_PREMAKNJEN	spremeni vrednost parametrov s in b	(ne uporablja)
NASLEDNJA_POVEZAVA	NASLEDNJA_POVEZAVA$_{\mathcal{SB}}$	NASLEDNJA_POVEZAVA$_{\mathcal{RSP}}$

Tabela 4.2-III: Nastavitve parametrov modela za algoritma $\mathcal{S}_4$ in $\mathcal{D}$

4.2.4 Kakovost usmerjevalnih algoritmov v dvokrožnih grafih

Statične usmerjevalne algoritme $\mathcal{S}_0$, $\mathcal{S}_1$, $\mathcal{S}_2$, $\mathcal{S}_3$ in $\mathcal{S}_4$ smo v okviru predhodnih raziskav implementirali v posebej za ta namen napisanem simulatorju. S pomočjo velikega števila simulacij usmerjanja (skupaj smo jih izvedli več kot 2.7 milijona) smo ocenili kakovost

algoritem $\mathcal{A}$	$K(\mathcal{A}, \mathcal{G}', \mathcal{P})$z
$\mathcal{S}_0$	1,986
$\mathcal{S}_1$	1,372
$\mathcal{S}_2$	1,348
$\mathcal{S}_3$	1,182
$\mathcal{S}_4$	1,117

Tabela 4.2-IV: Primerjava kakovosti usmerjevalnih algoritmov

posameznega algoritma. Tabela 4.2-IV prikazuje vrednost števila $K(\mathcal{A}, \mathcal{G}', \mathcal{P})$ za posamezen algoritem $\mathcal{A} \in \{\mathcal{S}_0,\mathcal{S}_1,\mathcal{S}_2,\mathcal{S}_3,\mathcal{S}_4\}$. Množica $\mathcal{G}'$ pri tem predstavlja unijo družin grafov $\mathcal{F}_1$ in $\mathcal{F}_2$, definiranih z

$$\mathcal{F}_1 = \{G(i;\ 1, s);\ i \in \{8, \ldots, 256\};\ \text{diam}(G(i;\ 1, s)) = \min_h \text{diam}(G(i;\ 1, h))\}$$

in

$$\mathcal{F}_2 = \{G(i;\ s_1, s_2);\ i \in \{8, \ldots, 256\};\ \text{diam}(G(i;\ s_1, s_2)) = \min_{h_1,h_2>1} \text{diam}(G(i;\ h_1, h_2))\},$$

$\mathcal{P}$ pa pravilo, ki danemu grafu G določi množico $\lfloor \log_2 n! \rfloor$ naključnih permutacijskih usmerjevalnih problemov (pri tem n predstavlja število vozlišč grafa G).
Na podlagi rezultatov omenjenih simulacij lahko naredimo naslednje zaključke. Vsaka sprememba v razvoju statičnih algoritmov je v povprečju izboljšala njihovo kakovost. Največje izboljšave sta prinesla ukinitev nepotrebnega čakanja po kriteriju sodo/liho (algoritem $\mathcal{S}_1$) in uvedba kriterija največje oddaljenosti pri reševanju trkov na povezavah (algoritem $\mathcal{S}_3$). Najboljši statični algoritem $\mathcal{S}_4$ je v povprečju manj kot 12% slabši od teoretično optimalnega algoritma (začetni algoritem $\mathcal{S}_0$ je od teoretično optimalnega algoritma slabši za skoraj 100%).

Algoritma $\mathcal{S}_4$ in $\mathcal{D}$
Ker je bil simulator, ki smo ga uporabili v preteklih raziskavah, namenjen le simulacijam v dvokrožnih grafih, smo v okviru dela doktorske disertacije izdelali nov, splošnejši program za simulacijo podatkovnega usmerjanja. Omenjeni simulator je napisan povsem v skladu z modelom usmerjevalnega algoritma, ki je predstavljen v tem delu (glej razdelek 4.2.1).
S tem simulatorjem smo primerjali kakovost algoritmov $\mathcal{S}_4$ in $\mathcal{D}$. Na grafih družin $\mathcal{F}_1$ in $\mathcal{F}_2$ smo izvedli 386.200 usmerjanj s prvim in enako število usmerjanj z drugim algoritmom. Na vsakem grafu omenjene družine smo rešili $\lfloor \log_2 n! \rfloor$ naključnih permutacijskih usmerjevalnih problemov. Ker je ta poskus za algoritem $\mathcal{S}_4$ identičen poskusu preteklih raziskav, ni presenetljivo, da se ujema tudi rezultat, ki se za opisano simulacijo glasi:

$K(\mathcal{S}_4)$=1,117. Bolj presenetljiv pa je rezultat, ki se nanaša na dinamičen algoritem – ugotovili smo namreč, da se je pri teh vhodnih podatkih algoritem $\mathcal{D}$ vedel zelo podobno kot algoritem $\mathcal{S}_4$. V 351.834 (91,1%) primerih sta algoritma končala v enakem številu usmerjevalnih korakov, v 16.645 primerih (4,3%) je bil boljši algoritem $\mathcal{S}_4$, v 17.721 primerih (4,6%) pa algoritem $\mathcal{D}$. Čeprav je algoritem $\mathcal{D}$ po teh testih rahlo boljši od algoritma $\mathcal{S}_4$, se to pozna šele v četrti decimalki števila K, saj velja $K(\mathcal{D})$=1,117.

4.3 Dinamično usmerjanje v k-krožnih grafih

Model usmerjevalnih algoritmov in dinamičen algoritem $\mathcal{D}$, ki sta v splošni obliki predstavljena v prejšnjih razdelkih, lahko uporabimo za implementacijo usmerjevalnega algoritma v poljubnem omrežju, ki ima naslednje tri lastnosti.

1. Vozlišča omrežja se med seboj razlikujejo samo po svojih oznakah.
Neformalno lahko to lastnost opišemo tudi na enega od naslednjih načinov: če pobrišemo oznake vozlišč, potem jih nikakor ne moremo več razlikovati; oziroma: vsako vozlišče je s preostalim omrežjem povezano na enak način kot vsa druga vozlišča; oziroma: poljubni dve vozlišči imata v omrežju popolnoma enako vlogo (t.j. nobeno vozlišče ni posebej odlikovano).

2. V omrežju je $2k$ tipov povezav.
Tipi povezav nastopajo v parih – vsakemu tipu pripada nasprotni tip. Učinek povezav nasprotnega tipa se vzdolž sprehoda izniči (t.j. končni cilj sprehoda se ne spremeni, če izločimo vse pare povezav nasprotnega tipa).

3. Vrstni red povezav vzdolž sprehoda ni pomemben.
Na cilj sprehoda pri dani začetni točki vpliva le število povezav posameznega tipa.

Prva lastnost je zelo splošna in jo pogosto zasledimo, saj jo imajo vsa omrežja z vozliščno simetrično topologijo. Drugi dve lastnosti sta bolj specifični, kljub temu pa obstaja velik razred tolopogij, ki ju premorejo. Vanj sodijo, med drugimi, tudi vsi Cayleyjevi grafi nad Abelovo grupo. Vprašanje o velikosti tega razreda navajamo v zaključku disertacije kot možnost za nadaljne raziskave.

Model usmerjevalnih algoritmov in algoritem $\mathcal{D}$ sta torej uporabna za usmerjanje v vseh omrežjih s topologijo Cayleyevih grafov nad Abelovo grupo (torej tudi za omrežja s topologijo k-krožnih grafov). Edini pogoj za uspešno delovanje algoritma v izbrani topologiji tega razreda je obstoj funkcije za izračun omejenih najkrajših poti. Ker sta oba, model in algoritem $\mathcal{D}$, že zapisana v splošni obliki, ju v tem razdelku ne bomo prepisovali, pač pa se bomo posvetili le delu, v katerem se posamezni algoritmi med seboj razlikujejo. Bralec lahko sam sestavi algoritem za splošen usmerjevalni problem v topologiji k-krožnih grafov tako, da uporabi model usmerjevalnih algoritmov, predstavljen na straneh 106–111, opis nastavitev za parametre modela v tabeli 4.2-III na strani 120, opis funkcije

NASLEDNJA_POVEZAVA$_{\mathcal{RSP}}$ na strani 103, in način za računanje omejene najkrajše poti iz nadaljevanja tega poglavja. Namesto permutacije VRSTNI_RED=$(2\ 1)$ iz tabele 4.2-III naj pri tem uporabi posplošeno permutacijo VRSTNI_RED=$(k\ k-1\ \ldots\ 1)$.

4.3.1 Omejene najkrajše poti v k-krožnih grafih

Računanje najkrajših poti v splošnih k-krožnih grafih je NP-težek problem [6]; ker lahko problem računanja najkrajših poti prevedemo na problem računanja omejenih najkrajših poti (RSP), je tudi problem RSP v splošnem primeru NP-težek. Pri fiksnem $k = 2$, pri katerem je velikost vhoda reda $\mathcal{O}(\log n)$ (glej razdelek 3.4.1), smo sicer pokazali, da se težavnost obeh problemov zmanjša – predstavili smo $\mathcal{O}(\log n)$ algoritma za reševanje obeh problemov – pri fiksnem $k > 2$ (tudi tu je velikost vhoda reda $\mathcal{O}(\log n)$) pa težavnost problemov še vedno ostaja odprto vprašanje (v [54] najdete domnevo, da je računanje najkrajših poti za ta primer NP-težek problem).

Hitrih algoritmov za reševanje RSP pri fiksnem $k > 2$ torej ne poznamo, poznamo pa preprost in zato tudi počasen algoritem, ki ga predstavimo v nadaljevanju tega razdelka. Ker je časovna zahtevnost tega algoritma $\mathcal{O}(n^k)$, torej eksponentna glede na velikost vhoda, je njegova uporaba omejena na grafe z manjšim številom vozlišč. Za grafe z velikim številom vozlišč pa smo razvili hiter $\mathcal{O}(\log n)$ algoritem za računanje približkov omejenih najkrajših poti – tudi ti so namreč uporabni pri reševanju usmerjevalnih problemov (pakete proti cilju vodimo po približkih najkrajših poti). Tudi algoritem za računanje približkov omejenih najkrajših poti predstavimo v nadaljevanju tega razdelka.

Predpostavili bomo, da je graf $G(n;\ \pm h_1, \ldots, \pm h_k)$ podan s številoma n in k ter vektorjem $h = (h_1, \ldots, h_k)$. Vse dvodimenzionalne objekte, ki smo jih uporabljali v poglavju 3, tu posplošimo na k-dimenzij: namesto mreže $\mathbb{Z}_2^2$ tu uporabljamo k-dimenzionalno mrežo $\mathbb{Z}^k$, namesto dvodimenzionalne baze tu uporabljamo k-dimenzionalno bazo in podobno. Omejeno najkrajšo pot bomo označili s $p_i^{(0,w)}$ ali natančneje z $RSP(n, k, h, w, i)$.

Algoritem s sistematičnim pregledovanjem

Prvi algoritem, ki ga bomo predstavili, je časovno zelo potraten, saj omejeno najkrajšo pot išče s sistematičnim pregledovanjem vseh ekvivalenčnih razredov po naraščajoči dolžini poti. Natančneje, algoritem v k-dimenzionalni celoštevilski mreži pregleduje enotske polkrogle (t.j. preseke enotskih krogel in i-tega polprostora) po naraščajočem polmeru. Za vsak element polkrogle preveri, ali ta predstavlja ekvivalenčni razred iskane omejene najkrajše poti; ko iskano pot najde, se ustavi.

V algoritmu 4.3-I smo z x označili element k-dimenzionalne mreže $\mathbb{Z}^k$, ki ga trenutno pregledujemo. Znak $\cdot$ v peti vrstici označuje skalarni produkt vektorja x in h – element x, za katerega velja $x \cdot h = x_1h_1 + x_2h_2 + \ldots + x_kh_k \equiv w \pmod{n}$, predstavlja pot med vozliščema 0 in w; če je x poleg tega še element polkrogle, predstavlja omejeno pot med 0 in w. Funkcija NASLEDNJI_NA_POLKROGLI(x, r, i) vrne element, ki je naslednik elementa x pri sistematičnem pregledovanju polkrogle $B_i(r)$, to je preseka krogle $B(r)$ (krogla v

Algoritem 4.3-I $kRSPS(n, k, h, w, i)$

Omejene najkrajše poti s sistematičnim pregledovanjem

```
kRSPS(n,k,h,w,i) :
 1:  r := 0; nasel:=false;
 2:  while not nasel do begin
 3:      r := r + 1; x := (r, 0, ..., 0);
 4:      repeat
 5:          if x · h ≡ w (mod n) then begin
 6:              nasel:=true;
 7:              break
 8:          end;
 9:          x = NASLEDNJI_NA_POLKROGLI(x, r, −i);
10:      until x = (0, 0, ..., 0);
11:  end;
12:  return x.
```

ℓ_1 metriki s polmerom r) in polprostora $\mathbb{Z}_i^k$. Ko funkcija NASLEDNJI_NA_POLKROGLI(x, r, i) pregleda celotno polkroglo (x nima naslednika), vrne vrednost $(0, 0, \ldots, 0)$. Funkcijo NASLEDNJI_NA_POLKROGLI smo v deveti vrstici algoritma klicali z argumentom $-i$, saj ob prepovedi povezav tipa i rešitev iščemo v polprostoru $\mathbb{Z}_{-i}^k$.
Sistematično pregledovanje kandidatov za omejene najkrajše poti se konča, ko algoritem najde prvi primerni element. To se zgodi najkasneje pri krogli s polmerom n. Skupno število pregledanih elementov je torej v najslabšem primeru $\mathcal{O}(n^k)$ – taka je tudi časovna zahtevnost algoritma 4.3-I.

Računanje približkov omejenih najkrajših poti

Drugi pristop k računanju omejenih najkrajših poti v k-krožnih grafih se zgleduje po algoritmu 3.13-I za računanje omejenih najkrajših poti v dvokrožnih grafih. Vse objekte, ki jih uporabljamo v tem algoritmu, smo posplošili in primerno prikrojili za uporabo v k-dimenzionalnem okolju; pri tem večjih težav ni bilo. Celo minimalna projekcija, ki je najpomembnejše orodje za iskanje rešitve, ima v k dimenzijah smiseln pomen, algoritem 3.9-III za njen izračun pa je še vedno dovolj hiter, da je njena uporaba upravičena.
Pakiranost baze modula X v primeru dvokrožnih grafov je glavni pogoj za uspešnost algoritma za računanje omejenih najkrajših poti – dokaz pravilnosti rezultata temelji prav na tej predpostavki. Žal v primeru k-krožnih grafov ne poznamo karakterističnih lastnosti za bazo, ki bi imela enake lastnosti kot pakirana baza v dveh dimenzijah, zato algoritem 4.3-II, ki ga bomo predstavili v nadaljevanju, uporablja "nepakirano" bazo k-dimenzionalnega

modula X in ni optimalen – rezultat, ki ga vrne, je omejena pot med danim parom točk, toda morda ne najkrajša. Kljub temu se je algoritem izkazal kot precej zanesljiv, saj, kot kažejo naše meritve, pravilen rezultat vrne z več kot 85% zanesljivostjo (rezultati meritev so podrobneje opisani na koncu razdelka).

Algoritem 4.3-II	$kRSPMP(n, k, h, w, i)$

Računanje približkov omejenih najkrajših poti

$kRSPMP(n, k, h, w, i)$:

A: $B :=$ baza k-dimenzionalnega modula X;

B: $x :=$ poljubna rešitev kongruenčne enačbe $x \cdot h \equiv w \pmod{n}$;

C: x_w := osnovni bazni element;

D: C_1^w := množica baznih elementov;

E: P := množica minimalnih projekcij elementov C_1^w vzdolž vektorjev B v $\mathbb{Z}_i^k$;

F: **return** $\min_{\ell_1} P$.

Sledi podrobna razlaga korakov algoritma 4.3-II za izračun približkov omejenih najkrajših poti ter ocena njegove časovne zahtevnosti.

A: Baza modula X. Ker želimo, da bi bil algoritem 4.3-II karseda podoben algoritmu, ki smo ga predstavili za dvodimenzionalni primer, moramo izbrati bazo modula X s podobnimi lastnosti, kot jih ima dvodimenzionalna pakirana baza. Žal karakterističnih lastnosti, ki bi k-dimenzionalni bazi zagotavljale, da je za vsak element znotraj osnovnega baznega paralelepipeda njegova razdalja do množice X dosežena v enem od oglišč tega paralelepipeda (ta lastnost je bistvena pri vseh izračunih v dvodimenzionalnem primeru), ne poznamo ([54]). Zato bomo uporabili drugo lastnost pakirane baze, namreč, da sta vektorja, ki jo sestavljata, po ℓ_1 normi najmanjša neodvisna elementa prostora X. Algoritem 4.3-III izračuna bazo B (t.j. matriko, katere stolpci so bazni vektorji $b_i,\ i = 1, \ldots k$) tako, da poišče prvih k neodvisnih elementov množice X – čeprav taka baza ni pakirana v smislu trditve 3.11-IV, je ta posplošitev največ, kar znamo v tem trenutku narediti.

Primer 4.3-I. Da baza, kot jo vrne algoritem 4.3-III, nima "temeljne lastnosti" (trditev 3.11-IV), ki bi omogočala hitro računanje najkrajših poti (in zaradi katere bi jo lahko imenovali *pakirana baza*), se prepričamo z naslednjim primerom. V grafu $G(12;\ \pm 1,\ \pm 4,\ \pm 5)$ so trije po ℓ_1 normi najmanjši elementi $\{\{0, 3, 0\}, \{1, 1, -1\}, \{2, 0, 2\}\}$. Razdalja točke $(2, 2, 0)$ do oglišč paralelepipeda, ki ga tvorijo ti vektorji, je 3, premer grafa $G(12;\ \pm 1, \pm 4, \pm 5)$ pa 2.

Algoritem 4.3-III bazo zgradi postopoma, s sistematičnim pregledovanjem krogel v ℓ_1 metriki; pri tem išče linearno neodvisne elementa modula X; algoritem se

Algoritem 4.3-III $KBaza(n,k,h)$

Baza k dimenzionalnega modula X

```
KBaza(n,k,h) :
 1:  r := 1; B := ∅;
 2:  while |B| ≠ k do begin
 3:     r := r + 1; x := (r,0,...,0);
 4:     repeat
 5:        if (x · h ≡ 0 (mod n)) and (rang(B ∪ {x}) = |B| + 1) then
 6:           B := B ∪ {x}
 7:        if |B| = k then break;
 9:        x := NASLEDNJI_NA_KROGLI(x, r);
10:     until x = (0,0,...,0);
11:  end;
12:  return B.
```

konča, ko najde k takih elementov. Sistematično pregledovanje opravi na podoben način kot algoritem 4.3-I – razlika med funkcijama NASLEDNJI_NA_POLKROGLI in NASLEDNJI_NA_KROGLI je le ta, da prva pregleda le polovico elementov (tiste, ki ležijo v preseku krogle in odgovarjajočega polprostora), druga pa vse. Preden algoritem 4.3-III nek element doda v bazo, se prepriča, če sta zanj izpolnjena dva pogoja (vrstica 5): z $(x \cdot h \equiv 0 \pmod n)$ preveri, če x leži v množici X, z $(\text{rang}(B \cup \{x\}) = |B| + 1)$ pa, ali je x neodvisen od elementov, ki so že v bazi. Časovna zahtevnost algoritma je $\mathcal{O}(n^k)$, saj mora v najslabšem primeru pregledati vse elemente v notranjosti krogle s polmerom n. Velikost bazne matrike je reda $\mathcal{O}(k^2)$, torej neodvisna od števila vozlišč omrežja. Ker je algoritem 4.3-III časovno precej potraten, je smiselno, da se bazna matrika izračuna enkrat za vselej ob konstrukciji omrežja; v vseh postopkih, ki to matriko potrebujejo, pa se predpostavi, da je dostop do nje omogočen v konstantnem času. Ker je k praviloma majhno število, je dodaten pomnilnik, ki ga za to porabimo, zanemarljiv.

B: Rešitev kongruenčne enačbe. Partikularno rešitev kongruenčne enačbe $x \cdot h \equiv w \pmod n$ poiščemo s k-kratno uporabo Evklidovega algoritma (glej razdelek 3.8). Časovna zahtevnost koraka B je torej $\mathcal{O}(k \log n)$.

C: Osnovni bazni element. Osnovni bazni element x_w izračunamo v $\mathcal{O}(k^3)$ (glej razdelek 3.8).

D: Bazni elementi. Bazne elemente dobimo z "elementarnimi" premiki osnovnega baznega elementa x_w v smereh baznih vektorjev. Pri $k = 2$ so to premiki v smereh 0, $-b_1$, $-b_2$ in $-(b_1+b_2)$, pri $k = 3$ pa premiki v smereh 0, $-b_1$, $-b_2$, $-b_3$, $-(b_1+b_2)$, $-(b_1+b_3)$, $-(b_2+b_3)$ ter $-(b_1+b_2+b_3)$. V splošnem velja, da so bazni elementi

tisti elementi množice X_w, ki jih dobimo, če od x_w odštevamo vsote elementov elementov potenčne množice množice baznih vektorjev.

Označimo z $\{0,1\}^k$ množico urejenih k-terk, t.j. elementov k-kratnega kartezičnega produkta $\{0,1\} \times \{0,1\} \times \ldots \times \{0,1\}$. Za vsak element $p \in \{0,1\}^k$, je $Bp = \sum_{i=1}^{k} p_i b_i$ en elementarni premik, zato je množica baznih elementov C_1^w enaka

$$C_w^1 = \{x_w - Bp;\ p \in \{0,1\}^k\}.$$

Časovna zahtevnost za izračun množice C_1^w je $\mathcal{O}(2^k)$.

E: Minimalne projekcije. Minimalno projekcijo v k dimenzionalen prostor izračunamo v času $\mathcal{O}(k)$ (glej razdelek 3.9.1). Ker je skupno število vseh potrebnih minimalnih projekcij enako $k2^k$, je časovna zahtevnost koraka E enaka $\mathcal{O}(k^2 2^k)$.

F: Rešitev naloge. Med največ $k2^k$ različnimi elementi najmanjšega poiščemo v $\mathcal{O}(k2^k)$ časa.

Skupna časovna zahtevnost algoritma 4.3-II je torej

$$\mathcal{O}(1) + \mathcal{O}(k \log n) + \mathcal{O}(k^3) + \mathcal{O}(2^k) + \mathcal{O}(k^2 2^k) + \mathcal{O}(k2^k).$$

Ker je število k fiksno in praviloma majhno ($k << n$), je časovna zahtevnost algoritma v praksi enaka $\mathcal{O}(\log n)$.

Kakovost približka

Da bi ocenili kakovost algoritma 4.3-II za računanje približkov omejenih najkrajših poti, smo omenjeni algoritem implementirali v programskem jeziku Mathematica. Na dveh družinah krožnih grafov smo z njegovo pomočjo izračunali približno 70.000 približkov. Rezultate smo primerjali z natančnimi vrednostmi, dobljenimi s pomočjo algoritma 4.3-I (tudi tega smo implementirali v Mathematici). Najprej smo algoritem preizkusili na družini 3-krožnih grafov $G(n;\ \pm 1, \pm 2, \pm 3)$, pri $n = 16, 17, \ldots, 100$. Z algoritmom 4.3-II smo izračunali 29.580 približkov omejenih najkrajših poti in ugotovili, da je bilo od tega 4.205 rezultatov (t.j. 14,2%) napačnih; povprečno je bila napačna pot za 1,64 (t.j. 25,1%) korakov daljša od prave. Podoben poskus smo naredili na družini 4-krožnih grafov $G(n;\ \pm 2, \pm 3, \pm 5, \pm 7)$, n=$16, 17, \ldots, 100$. Tu so bili rezultati malo slabši: čeprav je število napačnih rezultatov (relativno) manjše kot pri prejšnjem poskusu (od 39.440 izračunanih približkov je bilo 4.411 (t.j. 11,2%) napačnih), je odstopanje rezultatov od prave vrednosti večje (1,06 korakov oziroma 34,7%).

4.3.2 Kakovost usmerjevalnih algoritmov v k-krožnih grafih

S pomočjo simulatorja smo preverili delovanje algoritma $\mathcal{D}$ na nekaterih 2-, 3- in 4-krožnih grafih ter primerjali kakovost algoritmov pri reševanju posameznih problemov.

stopnja	$G(n;\pm2,\pm3)$		$G(n;\pm2,\pm3,\pm5)$		$G(n;\pm2,\pm3,\pm5,\pm7)$	
obremen.	$2-BF$	$2-MP$	$3-BF$	$3-MP$	$4-BF$	$4-MP$
nizka	1.08	1.08	1.09	1.36	1.14	1.65
srednja	2.86	2.86	2.46	2.69	2.38	2.98
visoka	5.13	5.13	4.21	4.63	3.94	4.99

Tabela 4.3-I: Vrednost števila $K(\mathcal{A})$ na testni množici

stopnja	$G(n;\pm2,\pm3)$		$G(n;\pm2,\pm3,\pm5)$		$G(n;\pm2,\pm3,\pm5,\pm7)$	
obremen.	$2-BF$	$2-MP$	$3-BF$	$3-MP$	$4-BF$	$4-MP$
nizka	6862	6862	4833	5987	4102	5826
srednja	18580	18580	11066	12078	8594	10782
visoka	33425	33425	19031	20897	14251	17959

Tabela 4.3-II: Število usmerjevalnih korakov posameznega algoritma

Za vsak $n = 16, 17, \ldots, 42$ smo problem z isto začetno razporeditvijo paketov v grafih $G(n;\ \pm2,\pm3)$, $G(n;\ \pm2,\pm3,\pm5)$ in $G(n;\ \pm2,\pm3,\pm5,\pm7)$ reševali z dvema različicama algoritma $\mathcal{D}$ – z algoritmom BF, ki uporablja natančne omejene najkrajše poti, dobljene z algoritmom 4.3-I, ter z algoritmom MP, ki uporablja približke omejenih najkrajših poti, dobljene z algoritmom 4.3-II. Vsak algoritem smo na posameznem grafu izvedli petdesetkrat, vsakič z drugo naključno začetno razporeditvijo paketov. Algoritme smo preizkusili pri treh različnih obremenitvah omrežja – v prvem primeru je bil v vsakem vozlišču na začetku en sam paket (nizka obremenitev omrežja), v drugem pet paketov (srednja obremenitev), v tretjem pa deset (visoka obremenitev). Rezultati simulacij so zbrani v tabelah 4.3-I in 4.3-II; prva prikazuje povprečno vrednost števila $K(\mathcal{A});\ \mathcal{A} \in \{2-MP, 2-BF, 3-MP, 3-BF, 4-MP, 4-BF\}$, druga pa skupno število usmerjevalnih korakov pri vseh 1.350 poskusih, ki smo jih s posameznim algoritmom opravili na opisani družini grafov.

Ujemanje prvega in drugega stolpca v obeh tabelah je povsem razumljivo, saj so približki za omejene najkrajše poti v primeru dvokrožnih grafov enaki natančnim vrednostim in torej algoritma 2-BF in 2-MP delata povsem enako.

Iz tabele 4.3-I vidimo, da je glede na vrednost cenilne funkcije K pri nizkih obremenitvah omrežja (prva vrstica tabele) najbolj primeren algoritem 2-MP. To razmerje se v prid algoritmov 3-BF, 3-MP in 4-BF obrne že pri prvem povečanju obremenitve omrežja (druga vrstica), algoritem 4-MP pa je boljši od 2-MP šele pri najvišji merjeni obremenitvi. Vendar iz teh rezultatov ne smemo prehitro sklepati, da algoritma 3-MP in 4-MP pri nizkih obremenitvah v praksi nista uporabna; iz tabele 4.3-II namreč vidimo, da sta v vseh primerih porabila bistveno manj korakov kot algoritem 2-MP. Seveda so algoritmi tipa BF mnogo primernejši (saj uporabljajo natančne vrednosti za najkrajše poti in ne približkov),

toda v primeru, ko je bolj kot skupno število usmerjevalnih korakov pomemben "reakcijski čas" (t.j. čas, ki ga algoritem porabi v vsakem usmerjevalnem koraku), je uporaba algoritmov 3-MP oziroma 4-MP povsem upravičena.
Zaključki zaradi majhnega vzorca niso povsem zanesljivi.

Poglavje 5

Zaključek

V prvem delu zaključka doktorske disertacije predstavimo avtorjeve izvirne prispevke k znanosti, v nadaljevanju pa nekatere možnosti za nadaljevanje dela.

5.1 Prispevki k znanosti

Poleg nekaterih znanih rezultatov iz literature so v tej doktorski disertaciji predstavljeni tudi naslednji avtorjevi izvirni prispevki k znanosti.

Topologija krožnih grafov.
Predstavil sem štiri različne definicije krožnih grafov, ki se pogosto pojavljajo v literaturi, in pokazal, da vse opisujejo isto družino. Zbral sem znane rezultate v povezavi s premerom, povezanostjo in simetrijo ter predstavil znane postopke za izračun premera, najkrajših poti in pakirane baze v dvokrožnih grafih. Opisal sem nekatere najbolj znane družine krožnih grafov ter podal nekaj grafičnih primerov.

Pakirana baza.
Izpeljal sem eksplicitno formulo (ne-iterativni postopek) za izračun pakirane baze dvokrožnih grafov z majhnih številom točk ter pokazal, kako lahko omenjeno formulo uporabimo za optimizacijo iterativnega postopka v splošnem primeru. Glede na lego baznih vektorjev sem pakirane baze razporedil v 11 skupin (tipi od A do K). Našel in dokazal sem nekatere lastnosti pakiranih baz, ki se nanašajo na smer diagonale baznega paralelograma (razlika baznih vektorjev) ter na karakteristične kote baznih vektorjev.

Minimalna projekcija in najmanjši element na traku.
Uvedel sem pojem minimalne projekcije, ki se nanaša na iskanje kratkih poti v krožnih grafih, in izpeljal formule za njen izračun v dvokrožnih in v splošnih k-krožnih ($k \geq 2$) grafih. Razvil in analiziral sem algoritem (računanje omejenih najkrajših poti v dvokrožnih grafih) v katerem minimalno projekcijo uporabljamo. Uvedel sem tudi pojem najmanjšega elementa na traku in podal algoritem za izračun v primeru $k = 2$.

Delno usmerjeni krožni grafi in omejene najkrajše poti.
Uvedel sem pojma delno usmerjenih k-krožnih grafov (to so grafi, v katerih so "skoraj" vse povezave neusmerjene, nekaj pa je usmerjenih) in omejenih najkrajših poti (to so najkrajše poti v delno usmerjenih grafih).

Algoritem za izračun omejenih najkrajših poti v dvokrožnih grafih.
Razvil sem učinkovit algoritem s časovno zahtevnostjo $\mathcal{O}(\log n)$ za računanje najkrajših poti v delno usmerjenih dvokrožnih grafih. Algoritem, katerega pravilnost sem formalno dokazal, je uporaben pri izvajanju učinkovitega dinamičnega usmerjanja v neusmerjenih dvokrožnih grafih, poleg tega pa je rezultat zanimiv tudi z matematičnega vidika, saj lahko algoritem uporabimo kot pripomoček za reševanje nekaterih linearnih diofantskih enačb z dodatnimi omejitvami.

Algoritma za izračun omejenih najkrajših poti v k-krožnih grafih.
Razvil sem $\mathcal{O}(n^k)$ algoritem za računanje omejenih najkrajših poti v splošnih k-krožnih grafih ($k \geq 2$) ter $\mathcal{O}(\log n)$ algoritem za računanje približkov teh poti. S primerjavo rezultatov obeh algoritmov na večjem številu testnih primerov sem ocenil kakovost algoritma za računanje približkov.

Model usmerjevalnih algoritmov.
Razvil sem enoten parametriziran zapis algoritmov za reševanje splošnega usmerjevalnega problema v k-krožnih grafih. Usmerjevalni algoritmi, predstavljeni v tem modelu, se med seboj razlikujejo le po nastavitvi nekaterih parametrov in jih zato lažje primerjamo. Poleg tega je implementacija algoritmov v simulatorju zelo preprosta, saj posamezen algoritem opišemo tako, da podamo pripadajoči nabor parametrov.

Simulator usmerjanja.
Razvil sem simulator usmerjanja v k-krožnih grafih, ki omogoča implementacijo algoritmov, skladno z opisanim modelom. Simulator sem uporabil za primerjavo kakovosti usmerjevalnih algoritmov.

Predstavitev statičnih algoritmov in zapis v modelu.
Predstavil sem pet statičnih algoritmov za reševanje splošnega usmerjevalnega problema v dvokrožnih grafih ter za vsakega od njih podal nabor parametrov, ki ga opisujejo v modelu usmerjevalnih algoritmov. Na podlagi rezultatov simulacij usmerjanja sem določil optimalno nastavitev parametrov modela za ta primer. Omenjene rezultate, skupaj s predstavitvijo modela, sem med drugim objavil v reviji *Journal of Systems Arhitecture*.

Razvoj dinamičnega usmerjevalnega algoritma.
Razvil in predstavil sem nov dinamični algoritem za reševanje splošnega usmerjevalnega problema v dvokrožnih grafih, ki uporablja optimalne nastavitve parametrov modela usmerjevalnih algoritmov, nadaljevanje poti v posameznem koraku pa določi s pomočjo algoritma za računanje omejenih najkrajših poti; prednost tega

algoritma pred statičnimi algoritmi je predvsem večja odpornost na napake. Dinamični algoritem sem predstavil v članku, ki je trenutno v recenziji v reviji *The Computer Journal*.

Primerjava kakovosti usmerjevalnih algoritmov v dvokrožnih grafih.
V simulatorju sem implementiral najboljši statični algoritem ter dinamični algoritem za usmerjanje v dvokrožnih grafih. Na podlagi večjega števila simulacij sem podal ugotovitve o kakovosti obeh algoritmov.

Posplošitev dinamičnega algoritma.
Dinamični usmerjevalni algoritem za usmerjanje v dvokrožnih grafih sem posplošil in podal v obliki, ki je primerna za usmerjanje v splošnih k-krožnih grafih ($k \geq 2$). Algoritem sem v simulatorju implementiral v dveh različicah – prva nadaljevanje poti na posameznem koraku določi na podlagi omejenih najkrajših poti, druga pa na podlagi približkov teh poti. Na podlagi rezultatov simulacij sem podal ugotovitve o kakovosti obeh različic algoritma.

5.2 Nadaljevanje dela

Oglejmo si še nekatera zanimiva vprašanja ter možnosti za nadaljevanje dela.

Posplošitev dinamičnega usmerjevalnega algoritma
V poglavju 4 smo predstavili model usmerjevalnih algoritmov in dinamični algoritem $\mathcal{D}$ za usmerjanje podatkov v dvokrožnih grafih ter pokazali, kako lahko oba posplošimo za uporabo v splošnih k-krožnih grafih. Še več, videli smo, da je posplošitev lahko tudi širša: model in algoritem $\mathcal{D}$ lahko brez večjih sprememb uporabimo na vseh Cayleyevih grafih nad Abelovo grupo. Zanimivo bi bilo ugotoviti, ali je to najširša mogoča posplošitev, to je, ali morda obstaja razred, večji od razreda Cayleyevih grafov, na katerega lahko posplošimo omenjene rezultate. Kot delni odgovor na to vprašanje navedimo domnevo, da lahko model usmerjevalnih algoritmov in tudi algoritem $\mathcal{D}$ brez večjih sprememb posplošimo na *razred abelskih krovnih grafov*, v katerega poleg krožnih sodijo tudi vsi Cayleyevi grafi nad Abelovo grupo, torusi in mnogi drugi. Zanimivo bi bilo poiskati tudi konkretne predstavnike omenjenih razredov, ki se pojavljajo v praksi.

Omejene najkrajše poti v k-krožnih grafih
Hitrost izvajanja algoritma $\mathcal{D}$ je odvisna od hitrosti izračuna omejenih najkrajših poti v izbrani topologiji. V disertaciji smo predstavili $\mathcal{O}(\log n)$ algoritem za izračun teh poti v dvokrožnih grafih ter omenili, da podobnega rezultata za k-krožne grafe ne poznamo. Obstaja celo domneva, da je iskanje teh poti v k-krožnih grafih NP-težek problem. Ali je to res? Če se bo izkazalo, da domneva drži, potem smo v zvezi z algoritmom $\mathcal{D}$ na področju k-krožnih grafov prišli do konca (v disertaciji je polinomski algoritem že opisan), če pa se bo izkazalo, da domneva ni pravilna, potem bi bilo dobro poiskati poli-logaritmičen algoritem tudi za k-krožne grafe.

Omejene najkrajše poti v drugih topologijah

Omenili smo, da je algoritem $\mathcal{D}$ mogoče posplošiti na veliko družino grafov (velikost te družine je še odprto vprašanje – glej prvo točko). Ker je hitrost algoritma $\mathcal{D}$ odvisna od hitrosti izračuna omejenih najkrajših poti, se odpira naslednja možnost za nadaljevanje dela: v družini grafov, na katere lahko posplošimo model usmerjevalnih algoritmov in algoritem $\mathcal{D}$, poišči take poddružine, da je za njene elemente iskanje omejenih najkrajših poti preprosta naloga (to je, da obstaja hiter algoritem). Poišči in opiši hitre algoritme za izračun omejenih najkrajših poti v izbrani topologiji (na primer za torus).

Pakirana baza v k dimenzijah

Pri razvoju algoritma za izračun omejenih najkrajših poti v dvokrožnih grafih smo videli, da je obstoj pakirane baze bistven. Ali tudi v k dimenzijah obstaja baza, s pomočjo katere bi lahko (po istem postopku kot v dveh dimenzijah) poiskali omejene najkrajše poti? Kakšne so karakteristične lastnosti take baze?

Eksplicitna formula za izračun pakirane baze

V disertaciji smo predstavili algoritem FPB, ki za dvokrožne grafe z majhnim številom točk ($n < 115$) pakirano bazo poišče brez iterativnega postopka (eksplicitna formula). Ali obstaja podoben rezultat tudi za splošne dvokrožne grafe? Če namreč taka formula obstaja, potem obstaja tudi eksplicitna formula za izračun premera dvokrožnega grafa (izračun tega temelji na izračunu pakirane baze).

Časovna zahtevnost računskih postopkov v odvisnosti od k

Upoštevajoč nekatere rezultate doktorske disertacije lahko postavimo naslednjo domnevo: z vsakim povečanjem kompleksnosti vhodnega problema (prehod iz usmerjenih na neusmerjene grafe ali prehod iz 2- na 3-krožne grafe) se bistveno spremeni časovna zahtevnost odgovarjajočega algoritma. Tako na primer za usmerjene dvokrožne grafe obstaja algoritem, ki v konstantnem času poišče omejene najkrajše poti [36], za neusmerjene dvokrožne grafe pa je časovna zahtevnost najboljšega nam znanega algoritma $\mathcal{O}(\log n)$. Podobno je pri računanju premera: za usmerjene dvokrožne grafe obstaja eksplicitna formula [8], najboljši nam znani algoritem za neusmerjene dvokrožne grafe je iterativni postopek s časovno zahtevnostjo $\mathcal{O}(\log n)$, za 3-krožne grafe pa obstaja celo domneva, da je problem računanja premera NP-težek. Ali so nam znani postopki optimalni in je torej res, da se z vsakim povečanjem kompleksnosti vhodnega problema pri krožnih grafih spremeni kompleksnost pripadajočega algoritma? Kako pa je s prehodom iz 3- na 4-krožne grafe?

Izjava o avtorstvu

Izjavljam, da sem doktorsko disertacijo z naslovom “Usmerjevalni algoritmi v omrežjih s topologijo krožnih grafov” izdelal samostojno pod vodstvom mentorja prof. dr. Boruta Robiča.

Tomaž Dobravec

Literatura

[1] F. Aguiló in M. A. Fiol. An efficient algorithm to find optimal double loop networks. *Discrete Mathematics*, 138:15–29, 1995.

[2] L.D. Aronson. Networks and routing - a survey. Tehnično poročilo 94-112, Faculty of Technical Mathematics and Informatics, Delft University of Technology, 1994.

[3] M. Baumslag in F. Annexstein. A unified framework for off-line permutation routing in parallel networks. *Math. Systems Theory*, 24:233–251, 1991.

[4] J.C. Bermond, F. Comellas, in D.F. Hsu. Distributed loop computer networks: A survey. *Journal of Parallel and Distributed Computing*, 24:2–10, 1995.

[5] A. Borodin. Towards a better understanding of pure packet routing. V *Proc. 3rd Workshop Alg. and Data Struct.*, številka 709 v Lect. Notes Comput. Sci, str. 14–25, 1993.

[6] J.Y. Cai, G. Havas, B. Mans, A. Nerurkar, J.P. Seifert, in I. Shparlinski. On routing in circulant graphs. V *Proceedings of the International Conference Computing and Combinatorics, COCOON'99, LNCS 1627*, str. 360–369, Tokyo, Japan, July 26–28 1999. Springer-Verlag, Berlin Heidelberg.

[7] R.C.F. Chan, C. Chen, in Z.-X. Hong. A simple algorithm to find the step of double-loop networks. *Discrete applied mathematics*, 121:61–72, 2002.

[8] Y. Cheng in F. K. Hwang. Diameters of weighted double loop networks. *Journal of Algorithms*, 9:401–410, 1988.

[9] C. Chou, D. J. Guan, in K. Wang. A dynamic fault-tolerant message routing algorithm for double-loop networks. *Information Processing Letters*, 70:259–264, 1999.

[10] B. Codenotti, I. Gerace, in S. Vigna. Hardness results and spectra techniques for combinatorial problems on circulant graphs. *Linear Algebra and its Applications*, str. 123–142, 1998.

[11] B. Codenotti, I. Gerace, in S. Vigna. On some combinatorial questions related to circulant graphs. *Linear Algebra and its Applications*, 285:123–142, 1998.

[12] G. Cooperman in L. Finkelstein. Permutation routing via cayley graphs with example for bus inteconnection networks. *DIMACS Series in Discrete Mathematics and Theoretical Computer Science*, 21, 1995.

[13] D. E. Culler in J. P. Singh with A. Gupta. *Parallel Computer Architecture; A Hardware/Software Approach*. Morgan Kaufmann, San Francisco, California, 1999.

[14] R. Cypher, F. Meyer auf der Heide, C. Scheideler, in B. Vöking. Universal algorithms for store-and-forward and wormhole routing. V *Proc. 28th ACM Symp. on Theory of Computing*, str. 356–365, 1996.

[15] P. J. Davis. *Circulant Matrices*. John Wiley & Sons, New York, 1979. Pure & Applied Mathematics; A Wiley - Interscience Series of Texts, Monographs & Tracts.

[16] T. Dobravec. *Permutacijsko usmerjanje v krožnih grafih: magistrska naloga*. november 2000.

[17] T. Dobravec in B. Robič. Računanje omejenih najkrajših poti z uporabo minimalne projekcije v omrežjih s topologijo 2-krožnih grafov. V *Dvanajsta elektrotehniška in računalniška konferenca ERK 2003*, str. 437–440, Ljubljana, Slovenija, 25.-26. september 2003.

[18] T. Dobravec, B. Robič, in B. Vilfan. Izmenjava podatkov v računalniških omrežjih. *Elektrotehniški vestnik*, 68(2-3):105–108, 2001.

[19] T. Dobravec, B. Robič, in B. Vilfan. Računalniška omrežja in usmerjevalni modeli. *Elektrotehniški vestnik*, 68(1):7–12, 2001.

[20] T. Dobravec, B. Robič, in B. Vilfan. Routing in double-loop networks. V *Proceedings of the IASTED International Symposia Applied Informatics*, str. 408–411, Innsbruck, Austria, February 2001.

[21] T. Dobravec, B. Robič, in B. Vilfan. Simulacija paketnega usmerjanja. V *Deseta elektrotehniška in računalniška konferenca ERK 2001*, str. B:59–62, Portorož, 24.-26. september 2001.

[22] T. Dobravec, B. Robič, in B. Vilfan. Dynamic shortest path routing in 2-circulants. V *Seventeenth International Symposium On Computer and Information Science, ISCIS XVII*, str. 332–336, Orlando, Florida, October 28-30 2002.

[23] T. Dobravec, B. Robič, in B. Vilfan. Najkrajše omejene poti v 2-krožnih grafih. V *Enajsta elektrotehniška in računalniška konferenca ERK 2002*, str. B:7–10, Portorož, Slovenia, 23.-25. september 2002.

[24] T. Dobravec, B. Robič, in B. Vilfan. Restricted shortest paths for routing in 2-circulant networks. V *IASTED International Conference, Communications and Computer Networks, CCN02*, str. 287–292, Cambridge, USA, November 4-6 2002.

[25] T. Dobravec, B. Robič, in J. Žerovnik. Izračun najkrajših poti v 2-krožnih grafih z uporabo pakiranih baz. V *Deveta elektrotehniška in računalniška konferenca ERK 2000*, str. A: 255–258, Portorož, 21.-23. september 2000.

[26] T. Dobravec, B. Robič, in J. Žerovnik. Permutation routing in double-loop networks: design and empirical evaluation. *Journal of Systems Architecture*, 48:387–402, 2003.

[27] T. Dobravec, J. Žerovnik, in B. Robič. An optimal message routing algorithm for circulant networks. (poslano v objavo).

[28] D. Z. Du, D. F. Hsu, Q. Li, in J. Xu. A combinatorial problem related to distributed loop networks. *Networks*, 20:173–180, 1990.

[29] D.S. Dummit in R.M. Foote. *Abrtract Algebra*. Prentice-Hall International, Inc., 1991.

[30] M. Escudero, J. Fiabrea, in P. Morillo. Fault-tolerant routing in double-loop networks. *Ars Combin.*, 25A:187–198, 1988.

[31] J. Fàbrega in M. Zaragozà. Fault-tolerant routing in double fixed-step network. *Discrete Applied Mathematics*, 78:61–74, 1997.

[32] M.A. Fiol, J.L.A. Yebra, I. Alegre, in M. Valero. A discrete optimization problem in local networks and data alignment. *IEEE Transactions On Computers*, C-36(6), 1987.

[33] G. Gauyacq, C. Micheneau, in A. Raspaud. Routing in recursive circulant graphs : edge-forwarding index and hamiltonian decomposition. V Juraj Hromkovic in Ondrej Sijkora, editors, *Graph-Theoretic Concepts in Computer Science, WG'98*, Lecture Notes in Computer Science, str. 227–241, 1998.

[34] M.D. Grammatikakis, D.F. Hsu, M. Kraetzl, in J.F. Sibeyn. Packet routing in fixed-connection networks: A survey. *Journal of Parallel and Distributed Computing*, 54(PC981483):77–132, 1998.

[35] J. Grasselli. *Diofantske enačbe*, zvezek 38 of *Knjižnica Sigma*. DMFA, 1984.

[36] D.J. Guan. An optimal message routing algorithm for double-loop networks. *Information Processing Letters*, 65:255–260, 1998.

[37] S.H. Huang, F.K. Hwang, in Y.H. Liu. Equivalent double-loop networks. *Taiwanese Journal of Mathematics*, 4(4):661–668, December 2000.

[38] F. K. Hwang in W. C. W. Li. Reliabilities of double-loop networks. *Probability in the Engineering and Information Science*, 5:255–272, 1991.

[39] F.K. Hwang. A complementary survey on double-loop networks. *Theoretical Computer Science*, 263:211–229, 2001.

[40] F.K. Hwang. A survey on multi-loop networks. *Theoretical Computer Science*, 299:107–121, 2003.

[41] F.K. Hwang, T.S. Lin, in R.H. Jan. A permutation routing algorithm for double loop network. *Parallel Processing Letters*, 7(3):259–265, 1997.

[42] Donald E. Knuth. *Fundamental Algorithms*, zvezek 1 of *The Art of Computer Programming*. Addison-Wesley, Reading, Massachusetts, third edition, 1997.

[43] V. Kumar, A. Grama, A. Gupta, in G. Karypis. *Introduction to Parallel Computing: Design and Analysis of Algorithms*. The Benjamin/Cummings, Redwood City, California, 1994.

[44] F.T. Leighton. *Introduction to Parallel Algorithms and Architectures*. Morgan Kaufmann Publishers, San Mateo, California, 1992.

[45] B. Mans. Optimal distributed algorithms in unlabelled tori and chordal rings. *Journal on Parallel and Distributed Computing*, 46:80–90, 1997.

[46] *O.G. Monakhov in E.A. Monakhova. Parallel systems with distributed memory: Structures and organization on interactions. Novosibirsk: SB RAS Publ. (in Russuan), 2000.

[47] P. Potočnik. *Delovanje grup in simetrije grafov : doktorska disertacija*. Univerza v Ljubljani, Fakulteta za matematiko in fiziko, 2000.

[48] C.S. Raghavendra, M. Gerla, in A. Avižienis. Peliable loop topologies for large local computer networks. *IEEE Transactions on Computer*, C-34(1):46–55, January 1985.

[49] B. Robič. Optimal routing in 2-jump circulant networks. Tehnično poročilo 397, University of Cambridge, U.K., Computer Laboratory, 1996.

[50] B. Robič in J. Žerovnik. Minimum 2-terminal routing in 2-jump circulant graphs. *Computers and Artificial Intelligence*, 19(1):37–46, 2000.

[51] M. Lovrečič Saražin. Caylejevi in petersenovi grafi. *Obzornik za matematiko in fiziko*, 42(6):193–198, 1995.

[52] H. J. Siegel. *Interconnection Networks for Large-Scale Parallel Processing*. McGraw-Hill, USA, second edition, 1990.

[53] T. Vidmar. *Računalniška omrežja in storitve*. Založba Atlantis, Ljubljana, 1997.

[54] J. Žerovnik in T. Pisanski. Computing the diameter in multiple-loop networks. *J. Algorithms*, 14:226–243, 1993.

[55] J. Žerovnik, B. Robič, in T. Dobravec. Optimal permutation routing in 2-jump circulant networks. V *Proc. ACIS 1st International Conference on Software Engineering Applied to Networking and Parallel/Distributed Computing SNPD*, str. 175–180, Reims, France, May 18-21 2000.

[56] R.J. Wilson in J.J. Watkins. *Uvod v teorijo grafov*. DMFA, Ljubljana, 1997.

*posredna referenca