Univerza v Ljubljani
Fakulteta za računalništvo in informatiko
Fakulteta za matematiko in fiziko

Miha Vučkovič

Računanje Fréchetove razdalje med krivuljama

diplomsko delo
na interdisciplinarnem univerzitetnem študiju
računalništva in matematike

Mentor: izred. prof. dr. Sergio Cabello Justo

Ljubljana 2013

Univerza *v Ljubljani*
Fakulteta *za računalništvo in informatiko*

Tržaška 25
1001 Ljubljana, Slovenija
telefon: 01 476 84 11
01 476 83 87
faks: 01 426 46 47
01 476 87 11
www.fri.uni-lj.si
e-mail: dekanat@fri.uni-lj.si

Št. naloge: 00042/2012
Datum: 05.11.2012

Univerza v Ljubljani, Fakulteta za računalništvo in informatiko ter Fakulteta za matematiko in fiziko izdaja naslednjo nalogo:

Kandidat: **MIHA VUČKOVIČ**

Naslov: **RAČUNANJE FRECHETOVE RAZDALJE MED KRIVULJAMA**
COMPUTING THE FRECHET DISTANCE BETWEEN CURVES

Vrsta naloge: Diplomsko delo univerzitetnega študija

Tematika naloge:

Frechetova razdalja med krivuljama je razdalja, ki upošteva parametrizacije krivulj. Alt in Godau (1995) sta opisala algoritem, ki izračuna razdaljo med poligonalnima potema. Rote (2007) je opisal algoritem za gladke krivulje. Cilj diplome je predstaviti Rotejev algoritem in napredne tehnike v algoritmu, kot je, na primer, parametrično iskanje, ter oceniti uporabnost Rotejevega algoritma.

Mentor:

izr. prof. dr. Sergio Cabello Justo

Fakulteta za računalništvo in informatiko • UNIVERZA V LJUBLJANI • 2

Dekan Fakultete za računalništvo in informatiko:

prof. dr. Nikolaj Zimic

Dekan Fakultete za matematiko in fiziko:

akad. prof. dr. Franc Forstnerič

UNIVERZA V LJUBLJANI • Fakulteta za matematiko in fiziko • 1

Izjava o avtorstvu diplomskega dela

Spodaj podpisani Miha Vučkovič, z vpisno številko **63050223**, izjavljam, da sem avtor diplomskega dela z naslovom:

Računanje Fréchetove razdalje med krivuljama

S svojim podpisom zagotavljam, da:

- sem diplomsko delo izdelal samostojno pod mentorstvom izred. prof. dr. Sergia Cabella Justa,
- so elektronska oblika diplomskega dela, naslov (slov., angl.), povzetek (slov., angl.) ter ključne besede (slov., angl.) identični s tiskano obliko diplomskega dela
- soglašam z javno objavo elektronske oblike diplomskega dela v zbirki "Dela FRI".

V Ljubljani, dne 23. januarja 2013 Podpis avtorja:

Zahvaljujem se mentorju prof. dr. Sergiu Cabellu za vso podporo in dragocene nasvete med pisanjem diplomske naloge. Hvala tudi kolegom in stricu Izidorju, ki so v času nastajanja nalogo prebrali in komentirali. Zadnja zahvala velja družini, ki mi je med študijem vedno stala ob strani.

Kazalo

Povzetek

Abstract

1 Uvod **1**

1.1 Pregled področja 2

1.2 Cilj naloge in rezultati 3

1.3 Organizacija dela 3

2 Metode za primerjanje podobnosti krivulj **5**

2.1 Hausdorffova metrika 5

2.2 Dinamično časovno krivljenje (angl. dynamic time warping) . . 6

2.3 Razdalja med krivuljami, ki temelji na odklonu 7

2.4 Fréchetova razdalja 8

3 Razumevanje Fréchetove razdalje **11**

4 Rešitev odločitvenega problema **15**

4.1 Predpogoji 15

4.2 Implementacija 22

5 Rešitev optimizacijskega problema **47**

5.1 Parametrično iskanje 47

5.2 Implementacija 51

6 Rezultati **57**

6.1 Odločitveni problem 57

6.2 Optimizacijski problem 59

7 Zaključek **61**

Povzetek

Primerjanje krivulj je v računalništvu pogost in pomemben problem. Krivulje se pogosto primerja kot diskretne množice točk, npr. z izračunom Hausdorffove razdalje. Tak pristop lahko vodi v neželene rezultate. Fréchetova razdalja, ki jo predstavimo v nalogi, je naravna metrika za podobnost krivulj, saj upošteva tako krivulje v celoti kot njihovo parametrizacijo. Obstaja tudi diskretna različica Fréchetove razdalje, ki prav tako upošteva tok krivulj, vendar ima v praksi precej hitrejše primitivne operacije.

Namen diplomskega dela je predstavitev razširitev algoritmov za izračun Fréchetove razdalje iz poligonalnih krivulj na gladke krivulje z določenimi omejitvami. V nalogi podamo konkretne algoritme za izračun odločitvenega in optimizacijskega problema Fréchetove razdalje za gladke krivulje pri podanih nekaterih primitivnih operacijah. Dokažemo, da ohranita kombinatorično časovno kompleksnost algoritmov, ki delujejo nad poligonalnalnimi krivuljami, t.j. $\mathcal{O}(n^2)$ za odločitveni problem ter $\mathcal{O}\big(n^2 \log^2 n\big)$ za optimizacijski problem, kjer sta krivulji sestavljeni iz n elementarnih delov.

Algoritme preizkusimo v praksi in ugotovimo, da imajo za praktično uporabo prepočasne primitivne operacije, zato je v praksi še vedno bolj smiselno uporabiti diskretno različico Fréchetove razdalje.

Ključne besede: krivulje, Fréchetova razdalja, parametrično iskanje

Abstract

Comparing curves is an important and common problem in computer science. Curves are usually compared as sets of points, for example using the Hausdorff distance. Such an approach can lead to unrealistic results. Fréchet distance, which we explain in this thesis, is a natural measure of similarity between curves because it uses the curves in their entirety and respects their parametrization. There exists a discrete variant of the Fréchet distance which also respects the flow of the curves. In practice, this variant has a much better performance because of simpler primitive operations.

The purpose of this thesis is to show the extension of algorithms for computing the Fréchet distance from polygonal curves to smooth curves, with some limitations. We provide concrete algorithms for solving the decision and optimization problems of Fréchet distance for smooth curves, provided some primitive operations are available. We prove that such algorithms make the same number of primitive operations as algorithms for polygonal curves: $\mathcal{O}(n)^2$ for the decision problem and $\mathcal{O}\big(n^2 \log^2 n\big)$ for the optimization problem, when the curves consist of n joined elementary pieces.

We conclude that the running times of the primitive operations of these algorithms are too large to be useful in practice, and that the discrete variant of the Fréchet distance is still a better option.

Keywords: curves, Fréchet distance, parametric search

Poglavje 1

Uvod

Problem primerjanja krivulj se v računalništvu pojavlja pogosto. Primerjanje je pomembno predvsem na področju računalniške grafike, računalniškega vida, robotike, analize signalov. Pri tem sta pomembni vprašanji:

- kako definiramo podobnost med krivuljama?
- kako učinkovito se da podobnost izračunati?

Poznanih je kar nekaj metrik in načinov primerjanja krivulj, za katere obstajajo učinkoviti algoritmi, vendar ne dajo nujno dobrih rezultatov. Nekaj takih metrik je opisanih v poglavju 2.

Fréchetova razdalja se imenuje po francoskem matematiku Maurice René Fréchetu in je metrika za primerjanje krivulj, ki v nasprotju z npr. Hausdorffovo metriko ne deluje na množici točk, ampak upošteva tudi tok krivulje. Fréchetova razdalja je bila pogosto uporabljena, recimo za učinkovito poenostavitev krivulj [1], iskanje po prostoročno pisanih dokumentih [20], aproksimacijsko iskanje najbližjega soseda [15] ter v geografskih informacijskih sistemih (GIS) [10].

Za boljšo predstavo si oglejmo naslednjo prispodobo o Fréchetovi razdalji planarnih krivulj f in g: naj pes stoji na začetku krivulje g, njegov lastnik pa na začetku krivulje f. Lastnik vodi psa na povodcu po krivulji g, sam pa se giblje po krivulji f, pri čemer se oba lahko premikata le naprej, dokler ne

prideta do končne točke svoje krivulje. Tedaj je Fréchetova razdalja najmanjša dolžina povodca, ki omogoča, da se pes in njegov lastnik premakneta od začetka do konca svojih krivulj pri navedenih omejitvah.

1.1 Pregled področja

Alt in Godau sta v [2] predstavila algoritem za izračun odločitvenega problema Fréchetove razdalje poligonalnih krivulj časovne zahtevnosti $\mathcal{O}(n^2)$, kjer sta poligonalni krivulji sestavljeni iz $\mathcal{O}(n)$ odsekov. Predstavila sta algoritem, ki Fréchetovo razdaljo za poligonalne krivulje izračuna v času $\mathcal{O}(n^2 \log n)$. Za izračun Fréchetove razdalje krivulj ob določenih dodatnih predpostavkah obstaja tudi $1+\varepsilon$ aproksimacijski algoritem skoraj linearne časovne zahtevnosti [11]. Algoritem za izračun preprostejše variante Fréchetove razdalje, diskretne Fréchetove razdalje, je bil predstavljen v [14] in ima časovno zahtevnost $\mathcal{O}(n^2)$.

Fréchetova razdalja se lahko razširi tudi na iskanje razdalje množice krivulj. Naivni algoritem za iskanje Fréchetove razdalje k krivulj ima časovno zahtevnost $\mathcal{O}\big(n^k \log n\big)$, poleg tega obstaja 2-aproksimacijski algoritem časovne zahtevnosti $\mathcal{O}(n^2 \log n)$ [12].

Obstajajo tudi algoritmi za izračun Fréchetove razdalje v bolj kompleksnih metričnih prostorih, recimo geodezična Fréchetova razdalja znotraj preprostih poligonov [16] in na konveksnih poliedrih [17]. Chambers *et al.* predstavijo drugo verzijo Fréchetove razdalje v večkotniku z luknjami, kjer tudi povodec predstavlja zvezna funkcija [9].

Fréchetovo razdaljo je možno razširiti na iskanje razdalje med ploskvami, recimo med preprostimi poligoni [8]. Buchin *et al.* so pokazali, da je odločitveni problem Fréchetove razdalje za preproste ne samo-sekajoče poligone z luknjami, in samo-sekajoče preproste poligone NP-težek [7].

Buchin *et al.* predstavijo algoritem za iskanje lokalno korektnih Fréchetovih prirejanj poligonalnih krivulj časovne zahtevnosti $\mathcal{O}(n^3 \log n)$ ter algoritem za iskanje diskretnega lokalno korektnega Fréchetovega prirejanja časovne

zahtevnosti $\mathcal{O}(n^2)$ [6].

Vsi znani algoritmi za izračun odločitvenega problema Fréchetove razdalje imajo, ne glede na dimenzijo prostora, v katerem sta krivulji definirani, skoraj kvadratično časovno zahtevnost, spodnja meja zahtevnosti tega problema pa je $\Omega(n \log n)$ [5].

1.2 Cilj naloge in rezultati

Rote je v [19] opisal, kako bi iskanje Fréchetove razdalje lahko razširili na splošne gladke krivulje, kar je bil povod za za pisanje naloge. Cilj naloge je bil predstaviti ter implementirati algoritem za iskanje Fréchetove razdalje zlepkov gladkih krivulj.

Predstavimo algoritem kombinatorične časovne zahtevnosti $\mathcal{O}(nm)$ za odločitveni problem Fréchetove razdalje zlepkov n in m gladkih krivulj ter algoritem kombinatorične časovne zahtevnosti $\mathcal{O}\big(nm \log^2(nm)\big)$ za optimizacijski problem.

Konkretno implementiramo in preizkusimo algoritem za odločitveni problem Fréchetove razdalje na primeru zlepkov, sestavljenih iz daljic ter krožnih odsekov. Predstavimo splošno ogrodje za implementacijo optimizacijskega algoritma Fréchetove razdalje, vendar ga zaradi kompleksnosti krivulj preizkusimo zgolj nad poligonalnimi krivuljami.

1.3 Organizacija dela

Struktura naloge je sledeča:

- V poglavju 2 opišemo nekaj različnih alternativ za določanje razdalj med krivuljami.
- V poglavju 3 formalno definiramo Fréchetovo razdaljo, diagrame krivulj ter množice prostih in prepovedanih točk diagramov. Definiramo tudi odločitveni in optimizacijski problem Fréchetove razdalje.

- V poglavju 4 opišemo potrebno teoretično ozadje ter algoritem odločitvenega problema Fréchetove razdalje nad zlepki gladkih planarnih krivulj.
- V poglavju 5 opišemo parametrično iskanje ter algoritem optimizacijskega problema Fréchetove razdalje nad zlepki gladkih planarnih krivulj.
- V poglavju 6 predstavimo rezultate konkretne implementacije za dva razreda krivulj: sklenjene daljice ter sklenjene krožne odseke. Opišemo probleme, ki se pri taki implementacijo pojavijo ter ideje za nadaljne raziskave.

Poglavje 2

Metode za primerjanje podobnosti krivulj

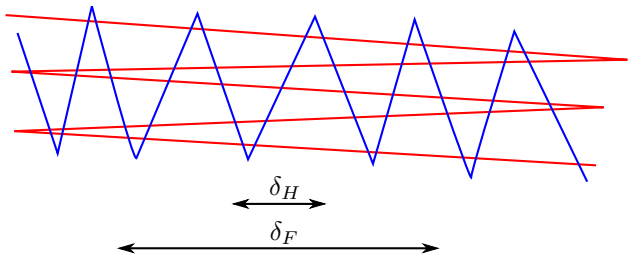
2.1 Hausdorffova metrika

Ena od možnosti računanja razdalj med krivuljami je *Hausdorffova metrika*. Kot vhod vzame dve neprazni množici točk in je definirana kot

$$\delta_H(A, B) = \max\left(\sup_{a\in A}\inf_{b\in B}\|a-b\|, \sup_{b\in B}\inf_{a\in A}\|a-b\|\right).$$

Hausdorffovo razdaljo lahko uporabimo za primerjanje podobnosti krivulj tako, da krivulje primerno vzorčimo, npr. v točkah presečišč. Ker Hausdorffova razdalja v izračun ne vključuje toka krivulje, saj jih obravnava kot množice točk, včasih ni najboljša možnost za določanje podobnosti krivulj. Tak primer prikazuje slika 6.1, ki je povzeta po [3].

Časovna zahtevnost za izračun Hausdorffove razdalje poligonalnih krivulj velikosti n, m delov je enaka $\mathcal{O}(nm)$.



Slika 6.1: Primer dveh krivulj z majhno Hausdorffovo razdaljo toda veliko Fréchetovo razdaljo.

2.2 Dinamično časovno krivljenje (angl. dynamic time warping)

Dinamično časovno krivljenje se je prvič pojavilo v 60. letih kot mera za primerjavo zvočnih zapisov. Podobno kot Hausdorffova razdalja tudi dynamic time warping deluje nad množicama točk. Naj bo $T = \{t_1, t_2, \ldots, t_n\}$ zaporedje časovnih vrednosti ter $A = \{a_1, a_2, \ldots, a_n\}$, $B = \{b_1, b_2, \ldots, b_n\}$ vrednosti funkcij pri vrednostih parametra t_i. Dinamično časovno krivljenje je definirano kot:

$$\delta_{DTW}(A, B) = \min_{\sigma} \sum_{(i,j)\in\sigma} \|b_j - a_i\|$$

kjer je $\sigma = \{\sigma_1, \sigma_2, \ldots, \sigma_n\}$, $\sigma_i \in [1, n]^2$ zaporedje, za katero velja:

- monotonost: za vse $(a, b), (c, d) \in \sigma$ kjer velja $a \leq c$ velja tudi $b \leq d$.
- zveznost: za vse $\sigma_i = (x_i, y_i)$, $\sigma_{i+1} = (x_{i+1}, y_{k+1})$ velja $x_{i+1} - x_i \leq 1$, $y_{i+1} - y_i \leq 1$.

Ker dinamično časovno krivljenje uporablja množice točk, je potrebno za njegovo uporabo pri primerjanju krivulj le-te prej vzorčiti, vendar pri tem upošteva tudi tok krivulj [13].

Časovna zahtevnost izračuna dinamičnega časovnega krivljenja poligonalnih krivulj velikosti n in m delov je enaka $\mathcal{O}(nm)$.

2.3 Razdalja med krivuljami, ki temelji na odklonu

Naj bo $f : [a, b] \to \mathbb{R}^2$ krivulja v ravnini. Naj bo $s : [a, b] \to \mathbb{R}$ funkcija dolžine loka krivulje f, definirana kot

$$s(t) = \int_a^t |f'(t)|\mathrm{d}t,$$

kjer $|f'(x)|$ označuje dolžino tangentnega vektorja krivulje f pri parametru t. *Odklon* (ang. turning angle) $\alpha_f : [a, b] \to \mathbb{R}$ krivulje f je definiran kot

$$\alpha_f(t) = \int_0^{s(t)} \kappa(s)\mathrm{d}s,$$

kjer je $\kappa(s)$ ukrivljenost krivulje f v odvisnosti od dolžine loka s, ki je definirana kot

$$\kappa = \frac{x'y'' - y'x''}{(x'^2 + y'^2)^{3/2}}.$$

Arkin *et al.* uvedejo metriko, ki temelji na primerjanju funkcij odklona krivulj [4]. Naj bo α_f funkcija odklona krivulje f. Potem lahko podobnost krivulj $f : [a, b] \to \mathbb{R}^2$ in $g : [a, b] \to \mathbb{R}^2$ določimo s primerjanjem funkcij odklonov α_f, α_g:

$$d(f, g) = \|\alpha_f - \alpha_g\| = \left(\int_a^b (\alpha_f(s) - \alpha_g(s))^2\mathrm{d}s\right)^{1/2}.$$

Ker je vrednost tako definirane $d(f, g)$ odvisna tudi od rotacije krivulj f in g, lahko rezultate še nekoliko izboljšamo:

$$\delta_T(f, g) = \min_{\theta\in\mathbb{R}} \left(\int_a^b (\alpha_f(s) - \alpha_g(s) + \theta)^2\mathrm{d}s\right)^{1/2}.$$

V primeru primerjanja dveh poligonalnih krivulj F, G velikosti n, m delov, v tem vrstnem redu, je časovna zahtevnost izračuna $\delta_T(F, G)$ enaka

$\mathcal{O}(nm \log(nm))$.

2.4 Fréchetova razdalja

Naj bosta $f : A = [l_A, r_A] \to \mathbb{R}^2$ in $g : B = [l_B, r_B] \to \mathbb{R}^2$ ravninski krivulji. *Zvezna Fréchetova razdalja* krivulj f in g je definirana kot

$$\delta_F(f, g) = \inf_{\substack{\alpha:[0,1]\to A \\ \beta:[0,1]\to B}} \max_{t\in[0,1]} \|f(\alpha(t)) - g(\beta(t))\|,$$

kjer sta α in β monotono naraščajoči funkciji, in velja $\alpha(0) = l_A$, $\alpha(1) = r_A$, $\beta(0) = l_B$, $\beta(1) = r_B$. Ko pogoj o monotonosti funkcij α in β umaknemo, govorimo o *šibki Fréchetovi razdalji*.

Časovna zahtevnost znanih algoritmov za problem odločanja za poligonalni krivulji F in G velikosti n in m delov, v tem vrstnem redu, je enaka $\mathcal{O}(nm)$, časovna zahtevnost znanih algoritmov za izračun točne Fréchetove razdalje pa je enaka $\mathcal{O}(nm \log(nm))$.

Naj bo $F : [0, n] \to V$ poligonalna krivulja. Definirajmo zaporedje $(F(0), F(1), \ldots, F(n))$ njenih točk z $\sigma(F)$. Naj bosta F in G poligonalni krivulji n in m odsekov, v tem vrstnem redu, in $\sigma(F) = (u_1, u_2, \ldots, u_n)$, $\sigma(G) = (v_1, v_2, \ldots, v_m)$ zaporedja njunih točk. *Prirejanje* L krivulj F in G naj bo urejeno zaporedje parov

$$(u_{a_1}, v_{b_1}), (u_{a_2}, v_{b_2}), \ldots, (u_{a_m}, v_{b_m}),$$

kjer velja $a_1 = 1, b_1 = 1, a_m = p, b_m = q$ in za vse $i \in [0, q]$ velja $a_{i+1} = a_i$ ali $a_{i+1} = a_i + 1$ in $b_{i+1} = b_i$ ali $b_{i+1} = b_i + 1$. *Diskretna Fréchetova razdalja* je definirana kot

$$\delta_{dF}(f, g) = \min_L \{ \max_{i\in[1,m]} \|u_{a_i} - v_{b_i}\| \},$$

kjer minimum teče po vseh možnih prirejanjih L.

Časovna zahtevnost izračuna diskretne Fréchetove razdalje poligonalnih krivulj F in G velikosti n in m odsekov, je enaka $\mathcal{O}(nm)$.

Fréchetova razdalja upoštevata tok krivulj, zato ima primer slike 6.1 veliko tako zvezno kot tudi diskretno Fréchetovo razdaljo.

Poglavje 3

Razumevanje Fréchetove razdalje

Za formalni opis problema izračuna Fréchetove razdalje najprej definirajmo nekaj novih pojmov.

Definicija 11.1. Definirajmo *diagram* $D_\varepsilon^{f,g} : A \times B \to \{-1, 0, 1\}$ krivulj $f : A \to \mathbb{R}^2$ in $g : B \to \mathbb{R}^2$ pri razdalji ε kot funkcijo

$$D_\varepsilon^{f,g}(t_f, t_g) = \begin{cases} 1 & ; \quad \|f(t_f) - g(t_g)\| < \varepsilon \\ 0 & ; \quad \|f(t_f) - g(t_g)\| = \varepsilon \\ -1 & ; \quad \|f(t_f) - g(t_g)\| > \varepsilon \end{cases}$$

Definicija 11.2. Definirajmo *množico prostih točk* diagrama $D_\varepsilon^{f,g}$ funkcij $f : A \to \mathbb{R}^2$, $g : B \to \mathbb{R}^2$ kot

$$F_\varepsilon(f, g) = \{(a, b) \in A \times B \mid D_\varepsilon^{f,g}(a, b) \in \{0, 1\}\},$$

množico prostih točk brez meje kot

$$F_\varepsilon^+(f, g) = \{(a, b) \in A \times B \mid D_\varepsilon^{f,g}(a, b) = 1\},$$

množico prepovedanih točk kot

$$F_\varepsilon^-(f,g) = \{(a,b) \in A \times B \mid D_\varepsilon^{f,g}(a,b) = -1\},$$

množico mejnih točk pa kot

$$F_\varepsilon^o(f,g) = \{(a,b) \in A \times B \mid D_\varepsilon^{f,g}(a,b) = 0\}.$$

Za točke znotraj F_ε rečemo, da so *dostopne* (angl. feasible), točke znotraj F_ε^- pa so *prepovedane* (angl. forbidden).

Definicija 12.1. Naj bo *ovira* O zaprta, povezana in maksimalna množica točk, za katero velja $O \subset F_\varepsilon^-(f,g)$. Množico vseh ovir v diagramu $D_\varepsilon^{f,g}$ označimo z $O_\varepsilon^{f,g}$.

V vseh ilustracijah so območja $F_\varepsilon^+(f,g)$ obarvana belo, območja $F_\varepsilon^-(f,g)$ sivo, množica točk $F_\varepsilon^o(f,g)$ pa je predstavljena s črno črto. Vse definirane pojme ilustrira slika 13.1.

Definicija 12.2. *Odločitveni problem* Fréchetove razdalje krivulj f in g je problem odločanja, ali za podano vrednost ε velja $\delta_F(f,g) \leq \varepsilon$.

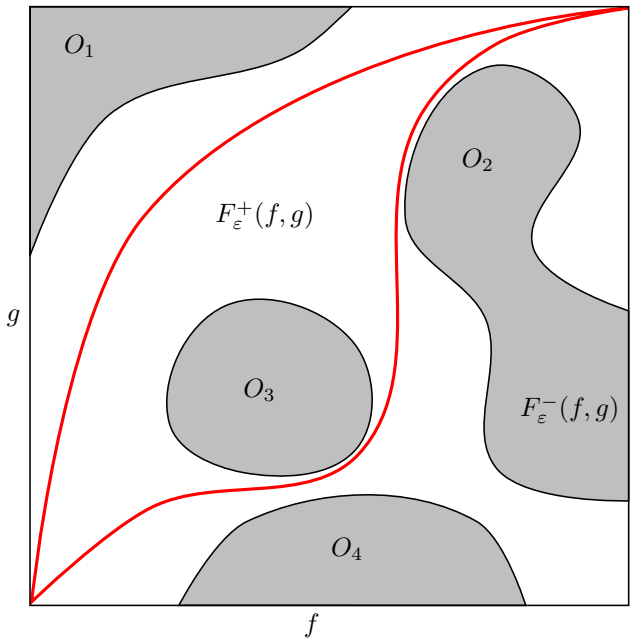
Definicija 12.3. Naj bosta $f : A = [l_A, r_A] \to \mathbb{R}^2$ in $g : B = [l_B, r_B] \to \mathbb{R}^2$ krivulji. Naj bo $p = (p_f : I \to A, p_g : I \to B)$ parametrična funkcija, kjer velja $\forall t \in I : p(t) \in F_\varepsilon(f,g)$. Potem je p *monotona pot* v diagramu $D_\varepsilon^{f,g}$, če velja

$$\forall t' < t : p_f(t') \leq p_f(t) \wedge p_g(t') \leq p_g(t).$$

Slika 13.1 prikazuje dve možni monotoni poti v diagramu.

Trditev 12.4. *Za nek $\varepsilon \geq 0$ velja $\delta_F(f,g) \leq \varepsilon$ natanko tedaj, ko v diagramu $D_\varepsilon^{f,g}$ obstaja monotona pot iz točke (l_A, l_B) v točko (r_A, r_B).*

Dokaz.
($\Rightarrow$) Po definiciji Fréchetove razdalje α in β predstavljata tako monotono pot



Slika 13.1: Ilustracija diagrama $D_\varepsilon^{f,g}$ krivulj f in g. Rdeči črti prikazujeta dve možni monotoni poti.

(v parametrični obliki) v diagramu $D_\varepsilon^{f,g}$.
($\Leftarrow$) Označimo s $p = (p_f : I \to A, p_g : I \to B)$ monotono pot v diagramu $D_\varepsilon^{f,g}$. Velja $\|f(p_f(t)) - g(p_g(t))\| \leq \varepsilon$ za vsako vrednost $t \in I$. Potem velja tudi

$$\max_{t\in I} \|f(p_f(t)) - g(p_g(t))\| \leq \varepsilon,$$

od koder sledi

$$\delta_F(f,g) = \inf_{\substack{\alpha:[0,1]\to A \\ \beta:[0,1]\to B}} \max_{t\in[0,1]} \|f(\alpha(t)) - g(\beta(t))\| \leq \max_{t\in I} \|f(p_f(t)) - g(p_g(t))\| \leq \varepsilon.$$

□

Definicija 13.1. *Optimizacijski problem* Fréchetove razdalje krivulj f in g je problem iskanja točne vrednosti $\delta_F(f,g)$.

V nadaljevanju naloge sledi natančen opis algoritmov za odločitveni in optimizacijski problem razširjenih na zlepke gladkih krivulj.

Poglavje 4

Rešitev odločitvenega problema

V tem razdelku bomo dokazali, da obstaja algoritem za odločitveni problem Frechetove razdalje zlepkov gladkih krivulj in ga konstruirali.

V podrazdelku 4.1 si bomo ogledali nekaj lem in omejitev, ki jih potrebujemo, da lahko algoritem implementiramo. Večina lem je povzetih po članku [19].

V podrazdelku 4.2 si bomo postopno ogledali algoritme, ki privedejo do končnega algoritma za odločitveni problem, in dokazali njihove časovne zahtevnosti. Konstruirali bomo algoritem za izračun ekstremnih točk ovir v diagramih, nato bomo definirali intervale v diagramih in ekvivalenčno relacijo med njimi. Odločitveni problem nato razdelimo na podprobleme iskanja dosegljivih točk znotraj enega diagrama. Na koncu vse algoritme združimo v algoritem odločitvenega problema Frechetove razdalje zlepkov gladkih krivulj n in m delov s časovno zahtevnostjo $\mathcal{O}(nm)$.

4.1 Predpogoji

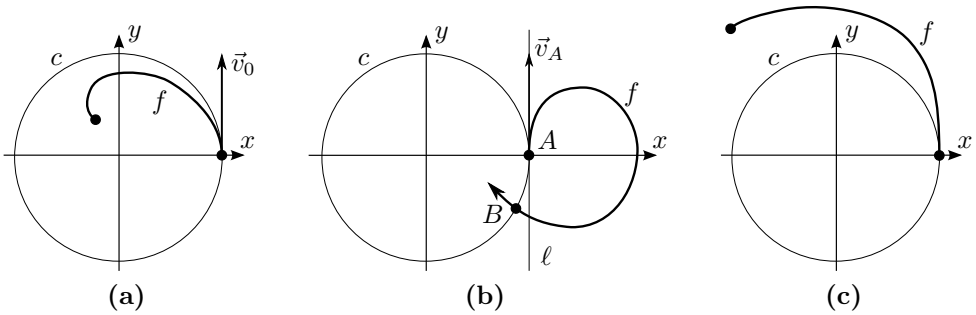
Lema 15.1. *Naj bo f dvakrat odvedljiva krivulja z odklonom največ π ter naj bo c krožnica polmera ε, funkcija f pa naj bo v enem od krajišč nanjo tangentna.*

(a) *Če je ukrivljenost f povsod strogo večja od $1/\varepsilon$ ter f in c tečeta v*

isto smer v začetni tangentni točki, potem krivulja f ne more doseči zunanjosti c.

(b) *Če je ukrivljenost f povsod strogo večja od $1/\varepsilon$ ter f in c tečeta v nasprotno smer v začetni tangentni točki, potem krivulja f ne more doseči notranjosti c.*

(c) *Če je ukrivljenost f povsod strogo manjša od $1/\varepsilon$ potem krivulja f ne more doseči notranjosti c.*



Slika 16.1: Ilustracije leme 15.1.

Dokaz.

(a) Brez škode za splošnost lahko predpostavimo, da je krog c enotski s središčem v točki $(0,0)$ začetna točka funkcije f je točka (1,0), tangentni vektor $\vec{v}_0$ pa kaže navzgor, kot prikazuje slika 16.1(a). Definirajmo f kot funkcijo dolžine loka s, $\alpha(s)$ pa naj predstavlja monotono naraščujočo funkcijo odklona funcije f. Označimo f ter njen odvod kot

$$f(s) = \begin{bmatrix} x(s) \\ y(s) \end{bmatrix},$$

$$\dot{f}(s) = \begin{bmatrix} \dot{x}(s) \\ \dot{y}(s) \end{bmatrix} = \begin{bmatrix} -\sin(\alpha(s)) \\ \cos(\alpha(s)) \end{bmatrix}. \tag{16.1}$$

Ker je $\alpha(s)$ po predpostavki monotono naraščujoča funkcija, velja $0 \leq \alpha(s) \leq \pi$ ter $\dot{\alpha}(s) > 1$. Potem velja $\dot{x} = -\sin(\alpha(s)) \leq 0$ ter

$$\dot{x} = -\sin(\alpha(s)) > -\sin(\alpha(s)) \cdot \dot{\alpha}(s).$$

Neenačbo lahko integriramo ter dobimo

$$x > \cos(\alpha).$$

Ker sta $\arccos(x)$ ter $\cot(x)$ monotono padajoči funkciji, za $-1 < x < 1$ dobimo

$$\begin{aligned} \arccos(x) &< \alpha \\ \frac{x}{\sqrt{1-x^2}} = \cot(\arccos(x)) &> \cot(\alpha) = \frac{\cos\alpha}{\sin\alpha}. \end{aligned} \tag{17.1}$$

Enačbo 16.1 in neenačbo 17.1 lahko združimo ter preuredimo v

$$\dot{y} < \frac{-x \cdot \dot{x}}{\sqrt{1-x^2}},$$

integriranje pa nam da končno mejo za y

$$y < \sqrt{1-x^2}.$$

(b) Predpostavimo situacijo iz točke (a). Glede na navpično premico l : $x = 1$ krožnica c leži na njeni levi, kot to prikazuje slika 16.1(b). Da bi krivulja f lahko prešla v notranjost krožnice c, bi morala sekali premico ℓ, s tem pa bi bil odklon glede na točko A večji od π, kar je v nasprotju s predpostavko.

(c) Dokažimo najprej, da trditev velja ob omejitvi $0 \leq \alpha(s) \leq \pi$. Brez škode za splošnost lahko predpostavimo enako situacijo, kot v točki (a), kot je prikazano na sliki 16.1(c).

Po predpostavki velja $|\dot{\alpha}(s)| < 1$. Potem velja $\dot{x} = -\sin(\alpha(s)) \leq 0$ ter

$$\dot{x} = -\sin(\alpha(s)) < -\sin(\alpha(s)) \cdot \dot{\alpha}(s).$$

Neenačbo lahko integriramo ter dobimo

$$x < \cos(\alpha).$$

Ker sta $\arccos(x)$ ter $\cot(x)$ monotono padajoči funkciji, za $-1 < x < 1$ dobimo

$$\begin{gathered} \arccos(x) > \alpha \\ \frac{x}{\sqrt{1-x^2}} = \cot(\arccos(x)) < \cot(\alpha) = \frac{\cos\alpha}{\sin\alpha}. \end{gathered} \tag{18.1}$$

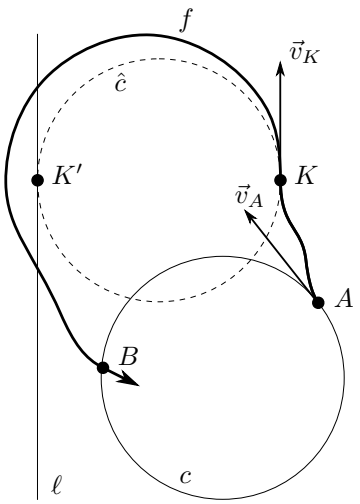
Enačbo 16.1 in neenačbo 18.1 lahko združimo ter preuredimo v

$$\dot{y} > \frac{-x \cdot \dot{x}}{\sqrt{1-x^2}},$$

integriranje pa nam da končno mejo za y

$$y > \sqrt{1-x^2}.$$

Sedaj odstranimo omejitev $0 \leq \alpha(s) \leq \pi$. Predpostavimo, da krivulja f seka krožnico c v točkah A in B. V točki K vrednost α doseže minimum, tangentni vektor $\vec{v}_K$ krivulje f pa v točki K kaže navzgor, kot to prikazuje slika 19.1. Ker trditev velja za $0 \leq \alpha(s) \leq \pi$, del krivulje f za točko K leži zunaj krožnice $\hat{c}$ polmera ε, ki je tangentna na vektor $\vec{v}_K$. Označimo navpično premico skozi skrajno levo točko K' krožnice $\hat{c}$ z ℓ. Ker je vrednost α v točki K po predpostavki minimalna, krožnica c gotovo v celoti leži desno od premice ℓ. Potem mora krivulja f sekati premico ℓ dvakrat, s tem pa bi bil odklon glede na točko K večji od π, kar je v nasprotju s predpostavko.



Slika 19.1: Splošna situacija leme 15.1(c).

□

Lema 19.1. *Naj bo f dvakrat odvedljiva funkcija z odklonom največ π, ter naj bo c krožnica polmera ε.*

(a) *Če je ukrivljenost f povsod strogo več od $1/\varepsilon$, potem f lahko seka c v največ dveh točkah.*

(b) *Če je ukrivljenost f povsod strogo manj od $1/\varepsilon$, potem f lahko seka c v največ dveh točkah.*

Dokaz.

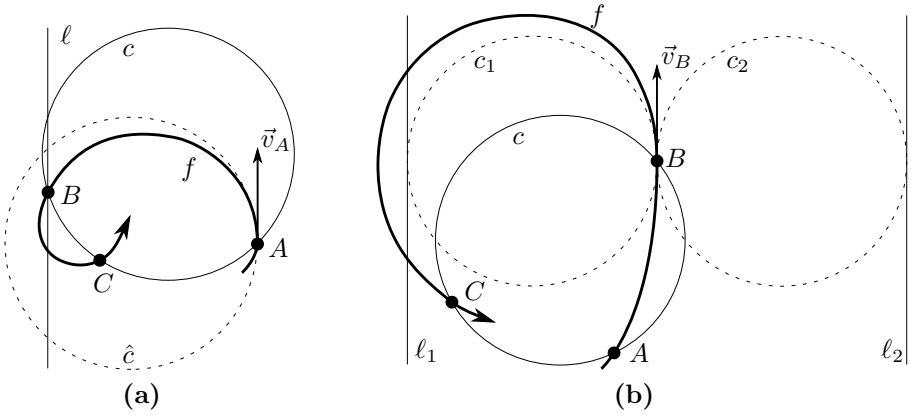
(a) Naj krivulja f seka krožnico c v točkah A, B, C. Predpostavimo lahko, da f v točki A preide v notranjost c, v točki B v zunanjost in v točki C nazaj v notranjost, tangentni vektor $\vec{v}_A$ krivulje f pa v točki A kaže navzgor, kot to prikazuje slika 20.1(a) (sicer orientacijo krivulje obrnemo). Po lemi 15.1 krivulja f leži v notranjosti krožnice $\hat{c}$ polmera ε, ki je tangentna na f v točki A. Da krivulja doseže točko C mora dvakrat sekati navpično premico ℓ skozi točko B, pri tem pa bi bil odklon glede na točko A večji od π, kar je v nasprotju s predpostavko.

(b) Naj krivulja f seka krožnico c v točkah A, B, C. Predpostavimo lahko, da f v točki A preide v notranjost c, v točki B v zunanjost in v točki C nazaj v notranjost, tangentni vektor $\vec{v}_B$ krivulje f pa v točki B kaže navzgor, kot to prikazuje slika 20.1(b) (sicer orientacijo krivulje obrnemo). S c_1, c_2 označimo krožnici polmera r, ki sta tangentni na krivuljo f v točki B, z B_1, B_2 njuni skrajni točki, z l_1, l_2 pa navpični premici skozi njiju. Potem mora krožnica c v celoti ležati med premicama ℓ_1 in ℓ_2, torej krivulja f seka eno od njiju dvakrat, s tem pa bi bil odklon glede na točko B večji od π, kar je v nasprotju s predpostavko.

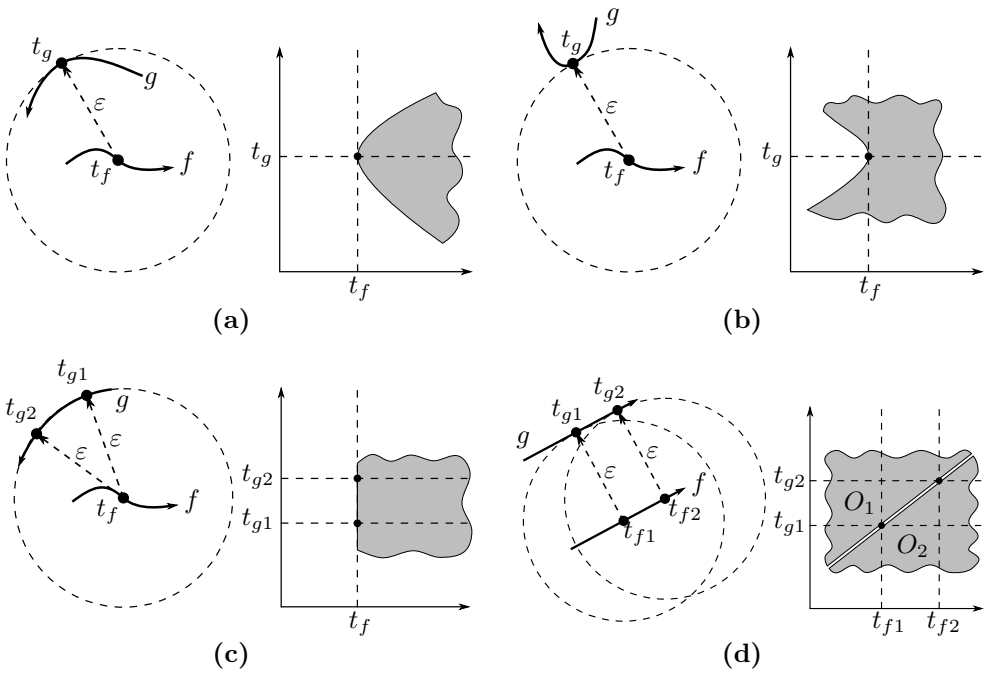
□

Lema 20.1. *Naj bosta $f : A \to \mathbb{R}^2$ in $g : B \to \mathbb{R}^2$ krivulji z ukrivljenostjo bodisi strogo več bodisi strogo manj od $1/\varepsilon$ ter odklonom največ π. Vsaka vodoravna ali navpična premica v diagramu $D_\varepsilon^{f,g}$ seka mejo $F_\varepsilon(f, g)$ največ dvakrat.*

Dokaz. Presečišča vodoravne premice z $F_\varepsilon^o(f, g)$ pri vrednosti $y = u$ so tedaj natanko presečišča krivulje f s krožnico polmera r in središčem v točki $g(u)$. Po lemi 19.1 sta presečišči največ dve. Enako velja tudi za navpično premico v diagramu. □



Slika 20.1: Ilustracija leme 19.1.



Slika 21.1: Ilustracija možnih situacij ekstremnih točk. Ilustracija (d) je shematična, oviri O_1 in O_2 se v točkah (t_{f1}, t_{g1}), (t_{f2}, t_{g2}) stikata.

Definicija 21.1. Naj bosta $f : A \to \mathbb{R}^2$ in $g : B \to \mathbb{R}^2$ krivulji z ukrivljenostjo bodisi vsaj $1/\varepsilon$ bodisi najmanj $1/\varepsilon$. Označimo s $C(\varepsilon, S) \subset \mathbb{R}^2$ krožnico polmera ε in središčem S, s $\dot{C}_X(\varepsilon, S) \in \mathbb{R}^2$ tangentni vektor take krožnice v točki X, z $\dot{g}(u)$ pa tangentni vektor krivulje g pri vrednosti parametra u.

Definirajmo množico ekstremnih točk diagrama kot

$$E_\varepsilon^{f,g} = \{(x, y) \in I \times J \mid g(y) \in C(\varepsilon, f(x)) \wedge \exists \lambda : \dot{C}_{g(y)}(\varepsilon, f(x)) = \lambda\, \dot{g}(y)\}.$$

Grafično lahko primere ekstremnih točk ponazorimo s sliko 21.1. Lahko se pojavita tudi dva robna primera: slika 21.1(c) ponazarja primer, ko krivulja g vsebuje krožni odsek polmera ε, krivulja f v tem primeru potuje skozi njegovo središče. Slika 21.1(d) ponazarja primer, ko sta dela krivulj f in g vzporedna ter je razdalja med njima ε.

Lema 21.2. *Naj bosta $f : A \to \mathbb{R}^2$ in $g : B \to \mathbb{R}^2$ krivulji z odklonom manjšim od π.*

(a) *Če je ukrivljenost krivulje g povsod strogo več ali strogo manj od $1/\varepsilon$, ima vsaka navpična premica v diagramu skozi katero izmed ekstremnih točk $e \in E_\varepsilon^{f,g}$ z množico $F_\varepsilon^o(f,g)$ le skupno točko e.*

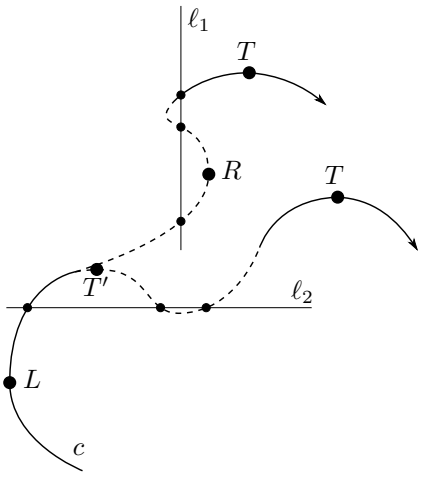
(b) *Če je ukrivljenost krivulje g povsod strogo več ali strogo manj od $1/\varepsilon$, ima vsaka vodoravna premica v diagramu skozi katero izmed ekstremnih točk $e \in E_\varepsilon^{g,f}$ z množico $F_\varepsilon^o(f,g)$ le skupno točko e.*

Dokaz. Dokažimo prvi primer, dokaz drugega primera je njemu analogen. Naj bo $(t_f, t_g) \in E_\varepsilon^{f,g}$ ekstremna točka. Tedaj je krivulja g v točki $g(t_g)$ tangentna na krožnico c polmera ε s središčem v točki $f(t_f)$. Krivuljo g v točki $g(t_g)$ razdelimo na dva dela. Za vsak tak del po lema 15.1 trdimo, da se krožnice c dotakne le v točki $g(t_g)$, torej se tudi celotna krivulja g krožnice c dotakne le v točki $g(t_g)$. Navpična premica v diagramu skozi (t_f, t_g) potem z množico $F_\varepsilon^o(f,g)$ res nima nobene druge skupne točke. □

Lema 22.1. *Naj krivulja c označuje mejo ovire O v $F_\varepsilon^-(f,g)$. Označimo najbolj levo (ne nujno ekstremno) točko z L, ter najvišjo (ne nujno ekstremno) točko ovire O z T. Potem med točkama L in T ni nobene ekstremne točke in c med L in T teče monotono v smereh x in y.*

Dokaz. Predpostavimo, da krivulja teče od točke L proti točki T, kot prikazuje slika 23.1. Potem se lahko iz točke L nadaljuje v desno ekstremno točko $R \in E_\varepsilon^{f,g}$, s čimer bi v neki točki navpična premica ℓ_1 sekala krivuljo c trikrat ali pa se nadaljuje v neko drugo zgornjo ekstremno točko $T' \in E_\varepsilon^{g,f}$, s čimer bi prav tako v neki točki vodoravna premica ℓ_2 sekala krivuljo c trikrat, kar je v protislovju z lemo 20.1. □

4.2 Implementacija



Slika 23.1: Ilustracija leme 22.1.

Definicija 23.1. Naj bodo $f_i : [t_{i-1}, t_i] \to \mathbb{R}^2$ gladke krivulje, kjer velja $t_0 \leq t_1 \leq \cdots \leq t_n$ ter $f_i(t_i) = f_{i+1}(t_i)$. Definirajmo *zlepek* krivulj f_i kot

$$F(t) = \begin{cases} f_1(t) & ; \quad t_0 \leq t \leq t_1 \\ f_2(t) & ; \quad t_1 \leq t \leq t_2 \\ & \vdots \\ f_n(t) & ; \quad t_{n-1} \leq t \leq t_n \end{cases}$$

Zlepke vedno pišemo z velikimi črkami, elementarne krivulje, ki jih sestavljajo, pa z malimi črkami. V algoritmih zlepke predstavimo z urejenimi množicami krivulj, uporabili pa bomo tudi naslednji oznaki za meji definicijskega območja elementarnih krivulj:

$$\min(f_i) = t_{i-1} \quad \max(f_i) = t_i.$$

Prevedimo nekaj definicij za splošne krivulje iz razdelka 3 na zlepke krivulj.

Definicija 23.2. Definicijo diagrama 11.1 prevedimo na zlepka F in G

$$D_\varepsilon^{F,G}(t_F, t_G) = \begin{cases} 1 & ; \quad \|F(t_F) - G(t_G)\| < \varepsilon \\ 0 & ; \quad \|F(t_F) - G(t_G)\| = \varepsilon \\ -1 & ; \quad \|F(t_F) - G(t_G)\| > \varepsilon \end{cases}$$

$D_\varepsilon^{F,G}$ imenujemo *mreža diagramov* zlepkov F, G.

Definicija 24.1. Definicije prostih, prepovedanih in mejnih točk 11.2 prevedimo na zlepek F krivulj $f_1, f_2, \ldots, f_n$ in zlepek G krivulj $g_1, g_2, \ldots, g_m$

$$F_\varepsilon(F, G) = \bigcup_{f_i, g_j} F_\varepsilon(f_i, g_j)$$
$$F_\varepsilon^+(F, G) = \bigcup_{f_i, g_j} F_\varepsilon^+(f_i, g_j)$$
$$F_\varepsilon^-(F, G) = \bigcup_{f_i, g_j} F_\varepsilon^-(f_i, g_j)$$
$$F_\varepsilon^o(F, G) = \bigcup_{f_i, g_j} F_\varepsilon^o(f_i, g_j)$$

Definiciji 23.2 in 24.1 prikazuje slika 25.1.

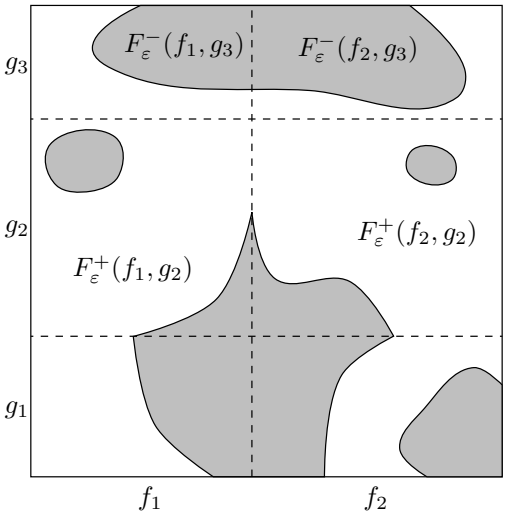
Definicija 24.2. Naj bo $f(t) = (x(t), y(t))$ parametrična ravninska krivulja. Definirajmo *vzporedni krivulji* f_ε^+ *in* f_ε^- krivulje f pri oddaljenosti ε kot

$$f_\varepsilon^+(t) = f(t) + \varepsilon \frac{(y'(t), -x'(t))}{\sqrt{x'^2(t) + y'^2(t)}}, \quad f_\varepsilon^-(t) = f(t) - \varepsilon \frac{(y'(t), -x'(t))}{\sqrt{x'^2(t) + y'^2(t)}}.$$

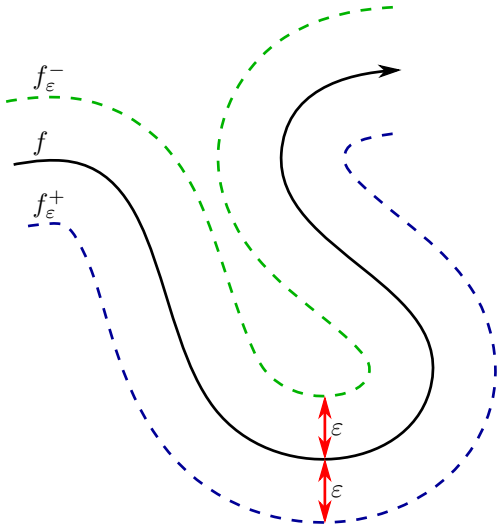
4.2.1 Predpriprava

Predpostavimo, da obstajajo naslednje primitivne operacije, ki so odvisne od opisa krivulj:

- CircleIntersections(f, S, ε): vrne množico $T \subset [\min(f), \max(f)]$ vrednosti parametrov krivulje f, kjer so presečišča krožnice polmera ε in središčem v točki S in krivulje f. Velja $|T| = \infty$ ali $|T| \leq 2$ (množica T je bodisi neskončna bodisi moči 2).



Slika 25.1: Ilustracija mreže diagramov $D_\varepsilon^{F,G}$ zlepka F krivulj f_1 in f_2 ter zlepke G krivulj g_1, g_2, g_3.



Slika 25.2: Ilustracija vzporednih krivulj f_ε^+ in f_ε^- krivulje f pri oddaljenosti ε.

- OffsetCurveIntersections(f, g, ε): Algoritem vrne množico parov (T_f, T_g), kjer $T_f \subset [\min(f), \max(f)]$ predstavlja množico parametrov krivulje f, $T_g \subset [\min(g), \max(g)]$ pa množico vrednosti parametrov vzporednih krivulj g_ε^+ in g_ε^- krivulje g, kjer je $(t_f \in T_f, t_g \in T_g)$ par parametrov presečišča krivulje f ter krivulje g_ε^+ ali g_ε^-. Predpostavimo, da glede na obliko krivulj f in g obstaja neka konstanta k, da je moč izhoda največ k.

- SplitAtCurvature(f, c): razdeli krivuljo f v vseh točkah t, kjer je ukrivljenost enaka c. Algoritem vrne množico delov krivulje f, pri katerih je ukrivljenost povsod večja od c, povsod manjša od c, ali pa povsod enaka c. Predpostavimo, da glede na obliko krivulje f obstaja neka konstanta k, da je velikost izhoda največ k.

- SplitAtCoefficient(f, c): razdeli krivuljo f v vseh točkah, kjer je naklon tangentnega vektorja enak c. Algoritem vrne množico takih delov krivulje f. Predpostavimo, da glede na obliko krivulje f obstaja neka konstanta k, da je moč izhoda največ k.

Naj bo F zlepek ravninskih krivulj $f_1, f_2, \ldots, f_n$ in G zlepek ravninskih krivulj $g_1, g_2, \ldots, g_m$, ε pa testna razdalja. Vsako od krivulj f_i, g_i lahko z zaporedno uporabo algoritmov SplitAtCoefficient ter SplitAtCurvature razdelimo na več delov tako, da je največji odklon vsakega dela največ π ter je ukrivljenosti vsakega dela povsod strogo večja od $1/\varepsilon$, strogo manjša od $1/\varepsilon$, ali pa enaka $1/\varepsilon$. Implementacijo predstavlja algoritem 26.1.

Ocenimo kombinatorično časovno zahtevnost in velikost izhoda algoritma 26.1. Zanka vrstice 3 se izvede n–krat, število obhodov zanke vrstice 4 pa je po predpostavki algoritma SplitAtCoefficient omejena s konstanto. Velikost izhoda je tako $\mathcal{O}(n)$, prav taka pa je tudi kombinatorična časovna zahtevnost.

Algoritem 26.1. Predpriprava.

Vhod: zlepek krivulj $F = \{f_1, f_2, \ldots, f_n\}$, koeficient k, testna razdalja ε
Izhod: množica predpripravljenih krivulj F'

```
1: procedure Preprocess(F, ε, k = 0)
2:     F' ← ∅
3:     for all f ∈ F do
4:         for all f' ∈ SplitAtCoefficient(f, k) do
5:             F' ← F' ∪ SplitAtCurvature(f', ε)
6:     return F'
```

V nadaljevanju predpostavimo, da sta $f : A \to \mathbb{R}^2$ in $g : B \to \mathbb{R}^2$ krivulji, katerih ukrivljenost je povsod strogo več od $1/\varepsilon$, strogo manj od $1/\varepsilon$, ali pa povsod enaka $1/\varepsilon$ ter njuna odklona manjša od π.

4.2.2 Izračun ekstremnih točk $E_\varepsilon^{f,g}, E_\varepsilon^{g,f}$

Trditev 26.1. *Označimo z $g_\varepsilon^+, g_\varepsilon^-$ vzporedni krivulji krivulje g, kjer je razdalja med g in g_ε^+ enaka ε ter razdalja med g in g_ε^- enaka ε. Velja*

$$E_\varepsilon^{f,g} = \{(x, y) \in A \times B \mid f(x) = g_\varepsilon^+(y) \vee f(x) = g_\varepsilon^-(y)\}.$$

Dokaz. Sledi iz definicije vzporedne krivulje. □

Trditev 26.1 nam omogoča, da množici ekstremnih točk $E_\varepsilon^{f,g}$ ter $E_\varepsilon^{g,f}$ lahko izračunamo z izračunom presečišč pripadajočih vzporednih krivulj. Pri tem uporabimo algoritem OffsetCurveIntersections.

Algoritem 27.1 predstavlja implementacijo izračuna ekstremnih točk $E_\varepsilon^{f,g}$. Ker nas zanimajo le ekstremne točke, od katerih so odvisne skrajne točke ovir, izmed štirih možnih situacij ekstremnih točk izločimo situacijo, ki jo prikazuje slika 21.1(d).

Ocenimo kombinatorično časovno zahtevnost in velikost izhoda algoritma 27.1. Ker je po predpostavki velikost izhoda OffsetCurveIntersections omejena s konstanto, ima zanka vrstice 3 konstantno število obhodov. Velikost izhoda algoritma 27.1 ter njena kombinatorična časovna kompleksnost sta torej $\mathcal{O}(1)$.

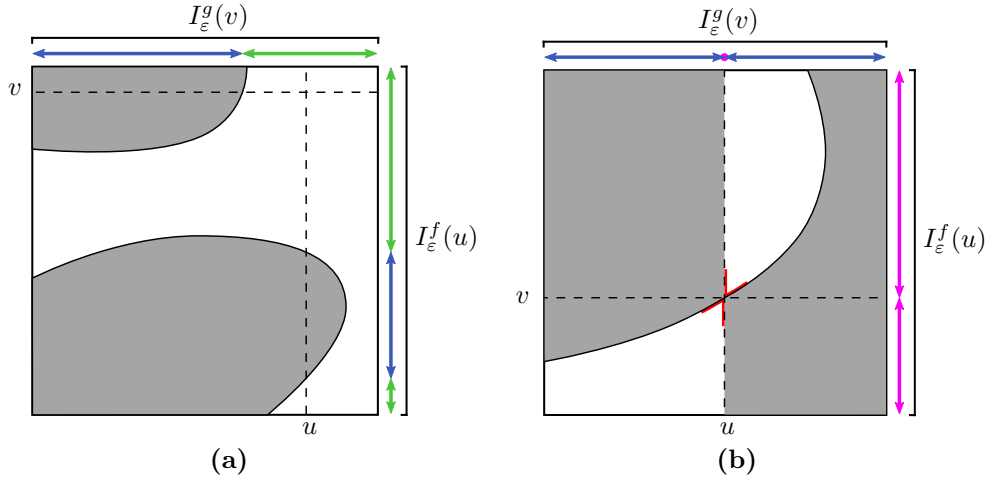
Algoritem 27.1. Izračun ekstremnih točk.

Vhod: krivulji f, g, testna razdalja ε
Izhod: množica ekstremnih točk $E_\varepsilon^{f,g}$

```
1: procedure ExtremePoints(f, g, ε)
2:     L ← ∅
3:     for all (t_f, t_g) ∈ OffsetCurveIntersections(f, g, ε) do
4:         # izločimo situacijo slike 21.1(d)
5:         if |t_f| = 1 or |t_g| = 1 then
6:             L ← L ∪ {(t_f, t_g)}
7:     return L
```

4.2.3 Izračun intervalov $I_\varepsilon^f(u)$

Definicija 27.1. Naj bo d daljica v diagramu $D_\varepsilon^{f,g}$ krivulj f in g s krajišči v točkah množice $F_\varepsilon^o(f,g)$ ali na stranicah diagrama. Tako daljico d, ki razen v krajiščih v celoti leži znotraj bodisi $F_\varepsilon^-(f,g)$ bodisi $F_\varepsilon^+(f,g)$, imenujemo *interval*.



Slika 28.1: Grafična predstavitev intervalov.

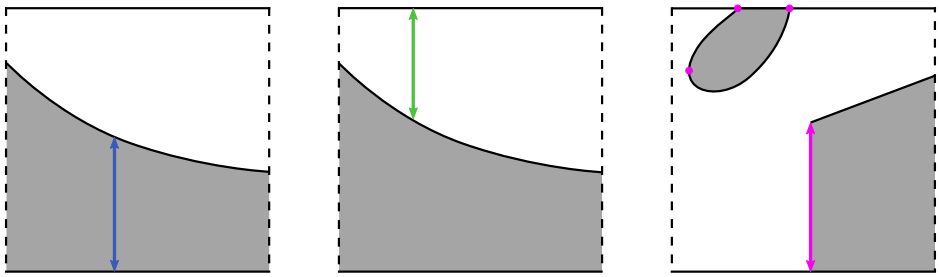
Za nas bodo pomembni zgolj navpični in vodoravni intervali diagrama, kakršne prikazuje slika 28.1.

Definicija 28.1. *Seznam intervalov* $I_\varepsilon^f(u)$ je urejena množica vseh intervalov na navpični premici diagrama $D_\varepsilon^{f,g}$ krivulj f in g skozi vrednost u parametra funkcije f. Seznam intervalov $I_\varepsilon^g(v)$ je urejena množica vseh intervalov na vodoravni premici diagrama skozi vrednost v parametra funkcije g.

Trditev 28.2. *Velja* $|I_\varepsilon^g(u)| \leq 3$ *ter* $|I_\varepsilon^f(u)| \leq 3$.

Dokaz. Direktna posledica leme 20.1. □

Definicija 28.3. Naj bo $J \in I_\varepsilon^g(u)$, $J = [a, b]$ interval. Definirajmo *tip*



Slika 29.1: Kategorizacija intervalov.

intervala J kot

$$\tau(J) = \begin{cases} & a \neq b \\ \oplus \;\; ; & \forall x \in (a,b) : D_\varepsilon^{f,g}(x,u) = 1 \\ & \forall x \in \{a,b\} : D_\varepsilon^{f,g}(x,u) = 0 \\ \\ \odot \;\; ; & \forall x \in [a,b] : D_\varepsilon^{f,g}(x,u) = 0 \\ \\ & a \neq b \\ \ominus \;\; ; & \forall x \in (a,b) : D_\varepsilon^{f,g}(x,u) = -1 \\ & \forall x \in \{a,b\} : D_\varepsilon^{f,g}(x,u) = 0 \end{cases}$$

Različni tipi intervalov so grafično predstavljeni na sliki 29.1. Iz definicije je razvidno, da je za določitev tipa $\tau(J)$ intervala $J = [a,b] \in I_\varepsilon^g(u)$ dovolj izračunati vrednost $\|f(x) - g(u)\|$ za poljubni $x \in (a,b)$. Ko velja $a = b$ je interval tipa $\odot$.

Naj se intervali tipa $\oplus$ imenujejo *prosti intervali* ter intervali tipov $\ominus$ in $\odot$ *ne-prosti intervali*.

Intervale $I_\varepsilon^f(u)$ pri vrednosti u parametra funkcije f lahko izračunamo, če poznamo presek premice skozi u ter množice $F_\varepsilon^o(f,g)$, kar so natanko presečišča krožnice polmera ε s središčem v točki $f(u)$ in krivulje g. Nekaj pazljivosti je potrebno, ko krivulja g sestoji iz krožnega odseka polmera ε, krivulja f pa potuje skozi njegovo središče, kot to prikazuje slika 21.1(c).

Takrat na premici skozi u leži en ali več intervalov tipa $\odot$. V primeru, ko je takih intervalov več, je na točki med prileţnimi pari intervalov zagotovo neka ekstremna točka $(t_g, u) \in E_\varepsilon^{g,f}$. Vrednost t_g tedaj predstavlja krajišči takih dveh intervalov. Tako ekstremno točko prikazuje točka (v, u) na sliki 28.1(b). Izračun intervalov $I_\varepsilon^f(u)$ predstavlja algoritem 30.1.

Ocenimo kombinatorično časovno zahtevnost in velikost izhoda algoritma 30.1. Ko je velikost izhoda algoritma CircleIntersections vrstice 2 končna, je velikost množice *ins* omejena s konstanto. V nasprotnem primeru ima zanka vrstice 5 konstantno število obhodov, torej je tudi v tem primeru velikost množice *ins* omejena s konstanto. Velikost izhoda in kombinatorična kompleksnost algoritma 30.1 je torej $\mathcal{O}(1)$.

Algoritem 30.1. Izračun intervalov.

Vhod: krivulji f, g, vrednost u parametra krivulje f, testna razdalja ε
Izhod: seznam intervalov $I_\varepsilon^f(u)$ v diagramu $D_\varepsilon^{f,g}$

```
1:  procedure Intervals(g, f, u, ε)
2:      ins ← CircleIntersections(g, f(u), ε)
3:      if |ins| = ∞ then        # funkcija g je krožni odsek radija ε
4:          ins ← ∅
5:          for all (t_g, t_f) ∈ ExtremePoints(g, f, ε) do
6:              if u ∈ t_f then
7:                  ins ← ins ∪ t_g
8:      ins ← Sort(ins ∪ {min(g), max(g)})
9:      I ← ∅
10:     for i ← 0; i < |ins| − 1; i ← i + 1 do
11:         I ← I ∪ {[ins[i], ins[i + 1]]}
12:     return I
```

4.2.4 Izračun kontrolnih parametrov

Definicija 30.1. Vrednost parametra c imenujemo *kontrolni parameter*, ko obstaja ovira $O \in O^{f,g}$, ki pri vrednosti parametra c doseže bodisi

skrajno točko bodisi seka stranico diagrama krivulj f in g. Označimo s $C^f = \{c_1, c_2, \ldots, c_\ell\}, c_1 \leq c_2 \leq \cdots \leq c_\ell$ množico urejenih kontrolnih parametrov funkcije f.

Iz definicije sledi, da so elementi množice C^f diagrama krivulj f, g vrednosti parametra funkcije f vseh ekstremnih točk množic $E_\varepsilon^{f,g}$ in $E_\varepsilon^{g,f}$ ter vrednosti parametra krivulje f vseh presečišč krivulje f s krožnicama polmera ε s središči v točkah $g(\min(g))$ ter $g(\max(g))$ (presečišča ovir s stranicami diagrama). Izračun kontrolnih parametrov je podan v algoritmu 31.1.

Ocenimo kombinatorično časovno zahtevnost in velikost izhoda algoritma 31.1. Velikosti množic ext_f, ext_g, *bottom*, *top* so po predpostavki omejene s konstanto. Od tod sledi, da imajo zanke vrstic 11, 13, 18 konstantno število obhodov, torej je velikost izhoda, pa tudi kombinatorična časovna zahtevnost $\mathcal{O}(1)$.

Algoritem 31.1. Izračun kontrolnih parametrov.

Vhod: krivulji f, g, meji a, b, testna razdalja ε

Izhod: urejen seznam parametrov $(\{c_1, d_1, \ldots, d_{\ell-1}, c_\ell\} \cap [a, b]) \cup \{a, b\}$, kjer so c_i, $a \leq c_i \leq b$, kontrolni parametri krivulje f, in $d_i \in (c_i, c_{i+1})$.

1: **procedure** ControlPoints(f, g, a, b, ε)
2: $ext_f \leftarrow$ ExtremePoints(f, g, ε),
3: $ext_g \leftarrow$ ExtremePoints(g, f, ε)
4: `# presečišča ovir na spodnji stranici`
5: $bottom \leftarrow$ CircleIntersections($f, g(\min(g)), \varepsilon$)
6: `# presečišča ovir na zgornji stranici`
7: $top \leftarrow$ CircleIntersections($f, g(\max(g)), \varepsilon$)
8: **if** $|bottom| = \infty$ **then** $bottom \leftarrow \emptyset$
9: **if** $|top| = \infty$ **then** $top \leftarrow \emptyset$
10: $L \leftarrow \emptyset, L' \leftarrow \emptyset$
11: **for all** $(t_f, t_g) \in ext_f$ **do**
12: $L \leftarrow L \cup \{t_f\}$
13: **for all** $(t_g, t_f) \in ext_g$ **do**

14: **if** $|t_f| = 1$ **then**
15: $L \leftarrow L \cup t_f$
16: `# vrednosti` c_i `na intervalu` $[a, b]$
17: $L \leftarrow (L \cup top \cup bottom) \cap [a, b]$
18: **for all** $c_i, c_{i+1} \in L$ **do**
19: $L' \leftarrow L' \cup \{c_i, (c_i + c_{i+1})/2, c_{i+1}\}$ `# dodaj vrednosti` d_i
20: **return** $L' \cup \{a, b\}$

4.2.5 Izračun skrajnih točk ovire O

Definicija 32.1. Definirajmo relacijo med ne-prostima intervaloma $I \in I_\varepsilon^g(u), J \in I_\varepsilon^g(v)$ kot

$$I \sim J \iff \exists O \in O^{f,g} : I \subset O \wedge J \subset O.$$

Ko sta u in v zaporedna kontrolna parametra in velja $u < v$ ter $I \sim J$, rečemo, da se *interval I preslika v interval J.*

Očitno je, da je taka relacija ekvivalenčna, elementi istega ekvivalenčnega razreda pa so intervali, ki pripadajo isti oviri O.

Trditev 32.2. *Naj bo $I \in I_\varepsilon^f(c_i)$ interval tipa $\ominus$, $I_\varepsilon^f(c_{i+1}) = \{J_1, J_2, \ldots, J_o\}$ pa množica intervalov, kjer sta $c_i, c_{i+1} \in C^f$ zaporedna kontrolna parametra. Potem lahko za $d_i \in (c_i, c_{i+1})$ in množico intervalov $I_\varepsilon^f(d_i) = \{K_1, K_2, \ldots, K_p\}$ določimo indeks j, da velja $I \sim K_j$ ter indekse $i_1, i_2, \ldots, i_r, r \leq 2$, da velja $J_{i_1} \sim J_{i_2} \sim \cdots \sim J_{i_r} \sim K_j$.*

Dokaz. Dokažemo s konstrukcijo. Predpostavimo, da je vsaj en interval množice $I_\varepsilon^f(c_i)$ tipa $\ominus$. Ločimo tri situacije, glede na tipe intervalov množic $I_\varepsilon^f(c_i), I_\varepsilon^f(d_i)$:

1. $\ominus$ ali $\oplus \rightarrow \ominus$ ali $\oplus$. Slika intervala tipa $\oplus$ je interval tipa $\oplus$, slika intervala tipa $\ominus$ pa je interval tipa $\ominus$. Velja $|I_\varepsilon^f(c_i)| = |I_\varepsilon^f(d_i)|$, torej so istoležni intervali množic $I_\varepsilon^f(c_i), I_\varepsilon^f(d_i)$ v relaciji.

2. $\ominus$, $\oplus$ ali $\odot \to \ominus$ ali $\oplus$. Izkaže se, da za vse take nabore tipov intervalov obstajajo enolične preslikave. Ker je naborov veliko, naj si bralec, ki ga zanima, natančne preslikave ogleda v programski kodi[1].

3. $\ominus$, $\oplus$ ali $\odot \to \ominus$, $\oplus$ ali $\odot$. Ker pri parametru d_i nobena od ovir ne doseže ekstremne točke, je edina možnost za nastanek te situacije, ko sta krivulji f in g vzporedni. Potem $I_\varepsilon^f(d_i)$ sestoji iz dveh intervalov tipa $\ominus$ in enega intervala tipa $\odot$, ki pripada neki ekstremni točki, in nima praslike. Tudi v tej situaciji velja $|I_\varepsilon^f(c_i)| = |I_\varepsilon^f(d_i)|$, torej so istoležni intervali množic $I_\varepsilon^f(c_i), I_\varepsilon^f(d_i)$ v relaciji.

□

Konstrukcijo zgornjega dokaza uporabimo za implementacijo algoritma FOLLOW(I, L), ki za podana $I \in I_\varepsilon^f(c_i), L = I_\varepsilon^f(d_i)$ ali $I \in I_\varepsilon^f(d_i), L = I_\varepsilon^f(c_{i+1})$ v konstantnem času izračuna intervale $J_i \in L$, da velja $J_i \sim I$.

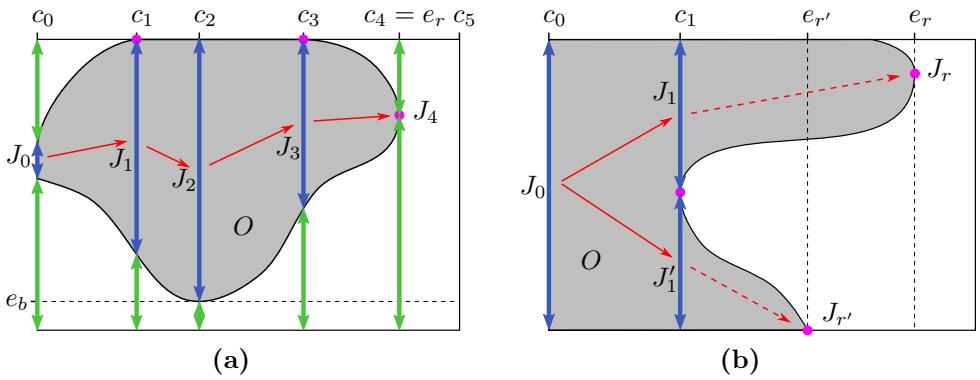
Ker se vse skrajne točke ovire O pojavljajo pri vrednostih kontrolnih parametrov, lahko trditev 32.2 uporabimo za konstrukcijo algoritma pometanja, s katerim izračunamo skrajne točke ovire O v treh smereh. Denimo, da iščemo zgornjo skrajno točko e_t, spodnjo skrajno točko e_b ter desno skrajno točko e_r ovire O na desni strani intervala J_0, ki tej oviri pripada v diagramu $D_\varepsilon^{f,g}$. Naj bo $\hat{C}^f = \{d_0, c_1, d_1, c_2, \ldots, c_\ell, d_\ell\}$, $d_0 = \min(f)$, $d_\ell = \max(f)$, $d_0 \le c_1$, $c_\ell \le d_\ell$, $\forall i \in [1, \ell - 1] : c_i < d_i < c_{i+1}$ množica urejenih parametrov funkcije f, kjer so $c_i \in C^f$ kontrolni parametri ter $I = \{I_\varepsilon^f(d_0), I_\varepsilon^f(c_1), I_\varepsilon^f(d_1), I_\varepsilon^f(c_2), \ldots, I_\varepsilon^f(c_\ell), I_\varepsilon^f(d_\ell)\}$ množica pripadajočih množic intervalov.

Iz začetnega intervala $J_0 \in I_\varepsilon^f(d_0)$ tipa $\ominus$ lahko izračunamo ne-proste intervale $J_i \in I_\varepsilon^f(c_i)$ ter $J_i' \in I_\varepsilon^f(d_i)$, dokler se ti ne končajo z intervalom J_ℓ' tipa $\ominus$, ko skrajna desna točka e_r ovire O ne obstaja, ali pa z intervalom J_k tipa $\odot$ pri kontrolnem parametru c_k, $1 \le k \le \ell$. V slednjem primeru pri parametru $c_k = e_r$ ovira O, ki določa ekvivalenčni razred $[J_i] = [J_i']$, doseže

[1]Programska koda je dostopna na `https://bitbucket.org/vucemz/frechet`.

skrajno desno točko. Vrednosti e_t in e_b določimo glede na meje intervalov J_i, J'_i.

Glede na trditev 28.2 se lahko ovira O razcepi na največ dva dela, kot to prikazuje slika 34.1(b). V tem primeru za oba dela ločeno poiščemo rešitve e_{r_i}, e_{t_i}, e_{b_i} ter jih naknadno združimo. Skrajno spodnjo točko določimo kot $e_b = \min_i(e_{b_i})$, zgornjo pa kot $e_t = \max_i(e_{t_i})$. Podobno kot za desno skrajno točko e_r ni nujno, da taka parametra e_t in e_b obstajata. Parameter e_t ne obstaja, ko se ovira O navidezno nadaljuje nad zgornjo stranico diagrama, kot je po prikazano na sliki 34.1(a). Podobno parameter e_b ne obstaja, ko se ovira O navidezno nadaljuje pod spodnjo stranico diagrama, parameter e_r pa ne obstaja, ko se ovira navidezno nadaljuje desno od desne stranice diagrama.



Slika 34.1: Uporaba trditve 32.2.

Izračun skrajnih točk ovire prikazuje algoritem 34.1.

Ocenimo kombinatorično časovno zahtevnost algoritma 34.1. Ker se po trditvi 32.2 ovira O razcepi na več delov v največ eni od kontrolnih točk, ima zanka vrstice 11 več kot en obhod v največ eni od vrednosti množice C. Velikost množice C se pri vsakem rekurzivnem klicu v vrstici 12 zmanjša, zato je število rekurzivnih klicev $\mathcal{O}(w)$, kjer je w začetna velikost množice C. Kombinatorična časovna zahtevnost algoritma 34.1 je torej $\mathcal{O}(w)$.

Algoritem 34.1. Izračun skrajnih točk ovire.

Vhod: krivulji f, g, urejena množica parametrov $C = \{v_0, v_1, \ldots, v_{w-1}\}$ krivulje g, kjer je množica C vsebovana v množici $\hat{C}^g$, interval $K_i \in I_\varepsilon^g(v_0)$ tipa $\ominus$, testna razdalja ε

Izhod: trojka (e_r, e_b, e_t) skrajno desne, zgornje in spodnje točke ovire O

1: **procedure** ObstacleExtremes($f, g, C, K_i, \varepsilon$)
2: $[a, b] \leftarrow K_i$
3: **if** $i > 0$ **then** $e_b \leftarrow a$ **else** $e_b \leftarrow -\infty$
4: **if** $i < r$ **then** $e_t \leftarrow b$ **else** $e_t \leftarrow \infty$
5: **if** $\tau(K_i) = \odot$ **then** $e_r \leftarrow v_0$
6: **else if** $|C| = 1$ **then** $e_r \leftarrow \infty$
7: **else**
8: $\{L_0, L_1, \ldots, L_r\} \leftarrow$ Intervals(f, g, v_1, ε)
9: $ind \leftarrow$ Follow($K_i, \{L_0, L_1, \ldots, L_r\}$)
10: $e_r \leftarrow -\infty$
11: **for all** $j \in ind$ **do**
12: $(e_{r_j}, e_{b_j}, e_{t_j}) \leftarrow$ ObstacleExtremes($f, g, C \setminus \{v_0\}, L_j, \varepsilon$)
13: $e_r \leftarrow \max(e_r, e_{r_j})$, $e_t \leftarrow \max(e_t, e_{t_j})$, $e_b \leftarrow \max(e_b, e_{b_j})$
14: **return** (e_r, e_b, e_t)

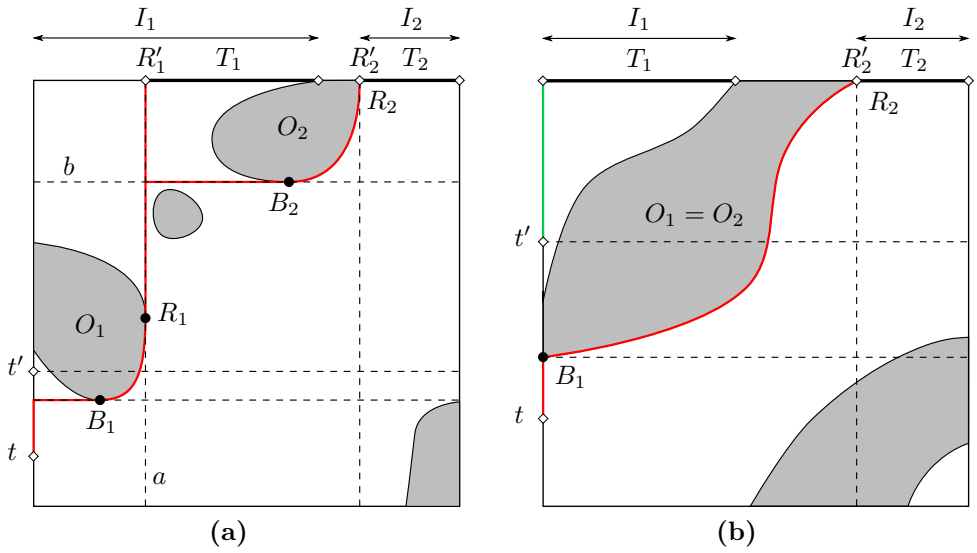
4.2.6 Izračun dostopnih točk na prileţni stranici diagrama krivulj

Naj bo $t \in F_\varepsilon(f, g)$ točka na levi stranici diagrama $D_\varepsilon^{f,g}$ krivulj f in g. Označimo oviro nad točko t z O_1. Po lemi 19.1 sta na zgornji stranici diagrama največ dva intervala prostih točk, označimo levi interval z I_1 ter desnega z I_2, oviro desno od intervala I_1 pa z O_2, kot to prikazuje slika 36.1. Ni nujno, da oviri O_1 in O_2 obstajata.

Lema 35.1. *Glede na pozicije ovir O_1, O_2 lahko konstruiramo monotono pot od točke t do množic dosegljivih točk T_1, T_2 na zgornji stranici $F_\varepsilon(f, g)$.*

Dokaz. Dokažimo s konstrukcijo takih dosegljivih intervalov T_1, T_2. Dovolj je, da konstruiramo najbolj levo dosegljivi točki R_1', R_2' na intervalih I_1, I_2, saj je

vsaka točka desno od točke R_i' na intervalu I_i direktno dosegljiva. Intervala T_1, T_2 potem lahko definiramo kot $T_i = \{x \in I_i \mid x \geq R_i'\}$.



Slika 36.1: Ilustracija konstrukcije dokaza leme 35.1.

V primeru, da vzporedna premica skozi t seka oviro O_1, je očitno, da sta intervala I_1, I_2 nedosegljiva. Tak primer predstavlja točka t' na sliki 36.1(a).

- Konstruirajmo najprej najbolj levo točko na intervalu I_1. Glede na obstoj ovire O_1 ločimo dva primera:

  1. ovira O_1 ne obstaja. V tem primeru je levi zgornji kot diagrama direktno dosegljiv iz točke t, to pa je tudi najbolj leva dosegljiva točka znotraj intervala I_1. Ta primer predstavlja točka t' na sliki 36.1(b).

  2. Ovira O_1 obstaja. Tedaj t leži pod njo. Ko velja $O_1 = O_2$, pot do intervala I_1 ne obstaja. Sicer lahko iz točke t dosežemo spodnjo točko B_1 ovire O_1. Taka točka B_1 je lahko tudi presečišče leve stranice diagrama z oviro, in ne nujno ekstremna točka. Ko je B_1 ekstremna točka, nam lema 21.2 zagotovi, da lahko do nje pridemo preko projekcije B_1 na levi stranici diagrama, kot je to nakazano z

rdečo črto na sliki 36.1(a). Lema 22.1 nam zagotovi, da od točke B_1 lahko po meji ovire O_1 dosežemo skrajno desno točko R_1 ovire O_1, ter njeno projekcijo R'_1 na zgornji stranici, ki je najbolj leva dosegljiva točka znotraj intervala I_1.

- Konstruirajmo še najbolj levo točko na intervalu I_2, če ta obstaja.

  Ko vzporedna premica skozi točko t seka oviro O_2, je očitno, da pot do intervala I_2 ne obstaja. Naj bo B_2 spodnja točka ovire O_2. Taka točka je zagotovo ekstremna, saj sicer interval prostih točk desno od ovire O_2 ne bi obstajal.

  Označimo navpično premico skozi skrajno desno točko R_1 ovire O_1 z a ter vodoravno premico skozi najnižjo točko B_2 ovire O_2 z b.

  Pokažimo, kako iz točke t lahko dosežemo točko B_2. Ločimo dva primera glede na obstoj ovire O_1:

  1. ovira O_1 ne obstaja. V tem primeru lahko s podobnim premislekom kot zgoraj dosežemo spodnjo točko B_2 ovire O_2.
  2. ovira O_1 obstaja. Točka t tedaj leži pod oviro O_1. Po lemi 19.1 mora biti točka B_2 zagotovo nad oviro O_1. Po lemi 21.2 lahko pridemo od desne točke R_1 ovire O_1 do presečišča premic a in b, od tu pa do točke B_2.

  Po meji ovire O_2 lahko pridemo od točke B_2 do skrajne desne točke R_2 ovire O_2, ter od tu do njene projekcije R'_2 na zgornji stranici diagrama, ki je najbolj leva dosegljiva točka znotraj intervala I_2.

□

Izrek 37.1. *Za podani gladki ravninski krivulji f in g obstaja algoritem, ki pri podani točki t na stranici diagrama $F_\varepsilon(f, g)$ v času $\mathcal{O}(1)$ izračuna množici T_1, T_2 dosegljivih točk iz točke t na njej priležni stranici diagrama.*

Dokaz. Dokažimo s konstrukcijo algoritma 38.1, pri tem uporabimo algoritem 34.1 za izračun skrajnih točk ovir ter konstrukcijo v dokazu leme 35.1.

Ocenimo kombinatorično časovno zahtevnost algoritma 38.1. Ker sta moči množici vrstic 3 in 4 omejeni s konstanto, lahko v konstantnem času najdemo indeksa i vrstic 6, 9, če ta obstajata. Velikost množice C vrstice 10 je prav tako konstantna, torej se algoritem ObstacleExtremes vrstice 11 izvede v $\mathcal{O}(1)$. Podobno velja tudi za vrstice 20–27. Kombinatorična časovna zahtevnost algoritma 38.1 je torej $\mathcal{O}(1)$.

Algoritem 38.1. Izračun začetnih točk dosegljivih intervalov priležne stranice.

Vhod: krivulji f, g, vrednost t parametra krivulje g, testna razdalja ε

Izhod: trojica (r_1, r_2, b), kjer sta r_1, r_2 vrednosti parametra funkcije f začetnih točk dosegljivih intervalov na priležni stranici, b pa dostopnost zgornje desne točke diagrama

1: **procedure** Adjacent(f, g, t, ε)
2: $r_1 \leftarrow$ null, $r_2 \leftarrow$ null, $b \leftarrow$ false
3: $\{L_0, L_1, \ldots, L_c\} \leftarrow$ Intervals($g, f, \min(f), \varepsilon$)
4: $\{T_0, T_1, \ldots, L_d\} \leftarrow$ Intervals($f, g, \max(g), \varepsilon$)
5: $tp \leftarrow \infty$
6: **if** exists i such that $\tau(T_i) = \ominus$ **then**
7: $[a, b] \leftarrow T_i, tp \leftarrow a$
8: # ovira O_1 obstaja
9: **if** exists i such that $\tau(L_i) = \ominus$ **and** $[a, b] \leftarrow L_i, t \leq a$ **then**
10: $C \leftarrow$ ControlPoints($f, g, \min(f), \max(f), \varepsilon$)
11: $(e_r, e_b, e_t) \leftarrow$ ObstacleExtremes($f, g, C, L_i, \varepsilon$)
12: # ovira O_1 sega pod točko t,
13: # ali pa sega do desne stranice diagrama $D_\varepsilon^{f,g}$
14: **if** $e_b < t$ **or** $e_r = \infty$ **then return** (r_1, r_2, false)
15: # ovira O_2, če obstaja, je desno od ovire O_1
16: **else if** $e_r < tp$ **then** $r_1 \leftarrow e_r$
17: # ovira O_2 je levo od desne točke ovire O_1

else $r_2 \leftarrow e_r$, **return** (r_1, r_2, true)

else $r_1 \leftarrow \min(f)$ `# ovira` O_1 `nad točko` t `ne obstaja`

if exists i such that $\tau(T_i) = \ominus$ **then** `# ovira` O_2 `obstaja`

$[a, b] \leftarrow T_i$

$\{M_0, M_1, \ldots, M_o\} \leftarrow \textsc{Intervals}(g, f, (a+b)/2, \varepsilon)$

$C_f \leftarrow \textsc{Reverse}(\textsc{ControlPoints}(f, g, \min(f), (a+b)/2, \varepsilon))$

$(e_{r_1}, e_{b_1}, e_{t_1}) \leftarrow \textsc{ObstacleExtremes}(f, g, C_f, M_o, \varepsilon)$

$C_f \leftarrow \textsc{ControlPoints}(f, g, (a+b)/2, \max(f), \varepsilon)$

$(e_r, e_{b_2}, e_t) \leftarrow \textsc{ObstacleExtremes}(f, g, C_f, M_o, \varepsilon)$

if $\textsc{Min}(e_{b_1}, e_{b_2}) \geq t$ **and** $e_r \neq \infty$ **then** $r_2 \leftarrow e_r$, $b \leftarrow \text{true}$

else $b \leftarrow \text{true}$ `# ovira` O_2 `ne obstaja`

return (r_1, r_2, b)

□

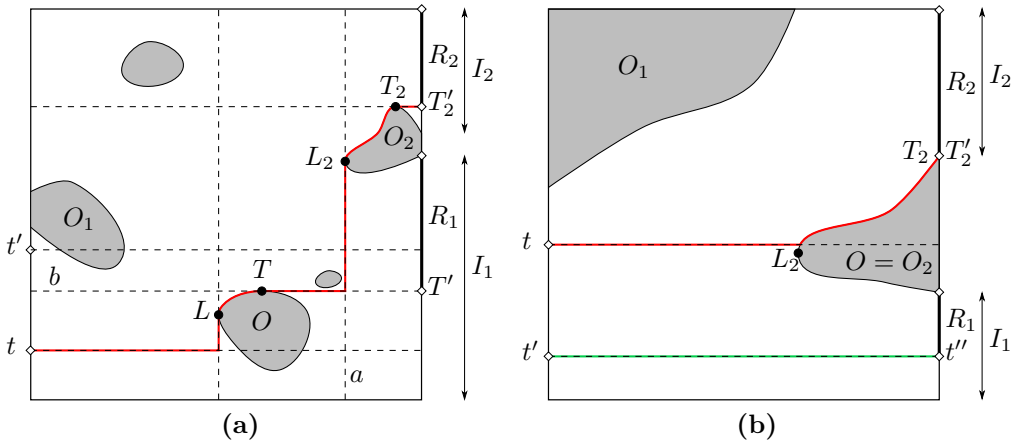
4.2.7 Izračun dostopnih točk na nasprotni stranici diagrama krivulj

Naj bo $t \in F_\varepsilon(f, g)$ točka na levi stranici diagrama $D_\varepsilon^{f,g}$ krivulj f in g. Označimo oviro nad točko t z O_1 ter prvo oviro, ki jo seka vodoravna premica skozi točko t z O. Po lemi 19.1 sta na desni stranici diagrama največ dva intervala prostih točk, označimo spodnji interval z I_1 ter zgornjega z I_2, oviro nad intervalom I_1 pa z O_2, kot to prikazuje slika 40.1. Ni nujno, da ovire O, O_1, O_2 obstajajo.

Lema 39.1. *Glede na pozicije ovir O_1, O_2, O_3 lahko konstruiramo monotono pot od točke t do množic dosegljivih točk R_1, R_2 na desni stranici $F_\varepsilon(f, g)$.*

Dokaz. Dokažimo s konstrukcijo takih dosegljivih intervalov R_1, R_2. Dovolj je, če konstruiramo najbolj levo dosegljivi točki T', T_2' na intervalih I_1, I_2. Intervala R_1, R_2 potem lahko definiramo $R_1 = \{x \in I_1 \mid x \geq T'\}$ in $R_2 = \{x \in I_2 \mid x \geq T_2'\}$.

V primeru, ko $O = O_1$, je očitno, da intervala I_1, I_2 nista dostopna.



Slika 40.1: Ilustracija konstrukcije dokaza leme 39.1.

- Konstruirajmo najnižjo dosegljivo točko na intervalu I_1.

  Ko je ovira O_2 pod točko t ali pa velja $O = O_2$, interval I_1 ni dosegljiv. Ko ovira O ne obstaja, je taka točka kar projekcija točke t na interval l_1. Primer take situacije je točka t' na sliki 40.1(b). Če O nima zgornje točke (sega do zgornje stranice diagrama), interval I_1 ni dosegljiv. Sicer ločimo dva primera glede na lokacijo presečišča vodoravne premice skozi točko t z oviro O:

  1. presečišče se nahaja pod skrajno levo točko L ovire O. Preko navpične premice skozi L lahko dosežemo točko L, kot to prikazuje slika 40.1(a). Od točke L lahko po lemi 22.1 po meji ovire dosežemo najvišjo točko T.
  2. presečišče se nahaja nad skrajno levo točko L ovire O. Od tu lahko po meji ovire O dosežemo najvišjo točko T direktno, kot to prikazuje slika 40.1(b).

  Najnižja dosegljiva točka na intervalu I_1 je projekcija T' točke T.

- Konstruirajmo še najnižjo točko na intervalu I_2, če ta obstaja. Ločimo dve situaciji:

1. ovira O ne obstaja, ovira O_2 pa leži pod točko t. Najnižja dosegljiva točka na intervalu I_2 je tedaj kar projekcija točke t.
2. ovira O obstaja. Ločimo dva primera glede na medsebojno pozicijo ovir O in O_2:
   (a) $O = O_2$. Takrat je najnižja točka na intervalu I_2 kar projekcija T_2' zgornje točke T_2 ovire O_2. Ta primer je prikazan na sliki 40.1(b).
   (b) Ovira O_2 leži nad oviro O. Označimo navpično daljico skozi skrajno levo točko L_2 ovire O_2 z a, vodoravno premico skozi najvišjo točko T ovire O pa z b. Po lemi 21.2 lahko pridemo od točke T do presečišča premic a in b, od tu pa do točke L_2. Po lemi 22.1 lahko pridemo od točke L_2 do zgornje točke T_2 ovire O_2, najnižja dosegljiva točka na intervalu I_2 pa je tedaj kar projekcija T_2' točke T_2.

□

Izrek 41.1. *Za podani gladki ravninski krivulji f in g obstaja algoritem, ki pri podani točki t na stranici diagrama $F_\varepsilon(f, g)$ v času $\mathcal{O}(1)$ izračuna množici T_1, T_2 dosegljivih točk iz točke t na njej nasprotni stranici diagrama.*

Dokaz. Dokažimo s konstrukcijo algoritma 41.1, pri tem uporabimo algoritem 34.1 za izračun skrajnih točk ovir ter konstrukcijo v dokazu leme 39.1.

Ocenimo kombinatorično časovno zahtevnost algoritma 41.1. Ker so moči množice vrstic 3, 4, 15 omejene s konstanto, lahko v konstantnem času najdemo indekse i vrstic 6, 10, 16. Velikost množice C vrstice 11 je omejena s konstanto, torej se algoritem ObstacleExtremes vrstice 12 prav tako izvede $\mathcal{O}(1)$. Podobno velja tudi za vrstice 17–24 in 31–37. Kombinatorična časovna zahtevnost algoritma 41.1 je torej $\mathcal{O}(1)$.

Algoritem 41.1. Izračun začetnih točk dosegljivih intervalov nasprotne stranice.

Vhod: krivulji f, g, vrednost t parametra krivulje g, testna razdalja ε

Izhod: trojica (r_1, r_2, b), kjer sta r_1, r_2 vrednosti parametra funkcije f začetnih točk dosegljivih intervalov na nasprotni stranici, b pa dostopnost zgornje desne točke diagrama

1: **procedure** OPPOSITE(f, g, t, ε)
2: $r_1 \leftarrow$ null, $r_2 \leftarrow$ null, $b \leftarrow$ false
3: $\{L_0, L_1, \ldots, L_c\} \leftarrow$ INTERVALS($g, f, \min(f), \varepsilon$)
4: $\{R_0, R_1, \ldots, R_d\} \leftarrow$ INTERVALS($g, f, \max(f), \varepsilon$)
5: $tp \leftarrow \infty$
6: **if** exists i such that $\tau(R_i) = \ominus$ **then**
7: $[a, b] \leftarrow R_i, tp \leftarrow a$
8: $e_{r'} \leftarrow \infty$
9: # ovira O_1 obstaja
10: **if** exists i such that $\tau(L_i) = \ominus$ **and** $[a, b] \leftarrow L_i, t \leq a$ **then**
11: $C \leftarrow$ CONTROLPOINTS($f, g, \min(f), \max(f), \varepsilon$)
12: $(e_{r'}, e_b, e_t) \leftarrow$ OBSTACLEEXTREMES($g, f, C, L_i, \varepsilon$)
13: **if** $e_b < t$ **then** # premica skozi t seka oviro O_1
14: **return** (r_1, r_2, false)
15: $\{M_0, M_1, \ldots, M_o\} \leftarrow$ INTERVALS(f, g, t, ε)
16: **if** exists i such that $\tau(M_i) = \ominus$ **then** # ovira O obstaja
17: $C_f \leftarrow$ CONTROLPOINTS($g, f, t, \max(g), \varepsilon$)
18: $(e_r, e_b, e_t) \leftarrow$ OBSTACLEEXTREMES($f, g, C_f, M_i, \varepsilon$)
19: # ovira O sega do vrha diagrama
20: **if** $e_r = \infty$ **then return** (r_1, r_2, false)
21: # ovira O_2 je nad e_r, če obstaja
22: **else if** $e_r \leq tp$ **then** $r_1 \leftarrow e_r$
23: # ovira O_2 je pod e_r, če obstaja
24: **else** $r_2 = e_r$, **return** (r_1, r_2, true)
25: # ovira O ne obstaja, ovira O_2 je nad t, če obstaja
26: **else if** $t \leq tp$ **then** $r_1 \leftarrow t$
27: # ovira O ne obstaja, ovira O_2 je pod t, če obstaja
28: **else** $r_2 \leftarrow t$, **return** (r_1, r_2, true)

29: `# ovira` O_2 `obstaja,` $O_1 \neq O_2$
30: **if** exists i such that $\tau(R_i) = \ominus$ **and** $e_{r'} \neq \infty$ **then**
31: $[a, b] \leftarrow R_i$
32: **if** $a \geq t$ **then**
33: $\{M'_0, M'_1, \ldots, M'_p\} \leftarrow$ INTERVALS$(f, g, (a + b)/2, \varepsilon)$
34: $C_f \leftarrow$ CONTROLPOINTS$(g, f, (a + b)/2, \max(g), \varepsilon)$
35: $(e_r, e_b, e_t) \leftarrow$ OBSTACLEEXTREMES$(f, g, C_f, M'_p, \varepsilon)$
36: **if** $e_r < \infty$ **then**
37: $r_2 \leftarrow e_r, b \leftarrow$ true
38: **else if** $e_{r'} \neq \infty$ **then** $b \leftarrow$ true `# ovira` O_2 `ne obstaja`
39: **return** (r_1, r_2, b)

□

4.2.8 Odločitveni problem

Pokažimo, kako za dani diagram krivulj f, g in dana para $(l_1, l_2), (b_1, b_2)$ dosegljivih točk na levi in spodnji stranici diagrama izračunamo dostopne točke na zgornji in desni stranici. Algoritma 38.1 in 41.1 nam omogočita, da za eno dosegljivo točko na levi ali spodnji stranici izračunamo želena dosegljiva intervala. Algoritma uporabimo na vsaki od štirih točk l_1, l_2, b_1, b_2, za končni rezultat pa izberemo minimum pripadajočih rešitev, kot je prikazano v algoritmu 43.1.

Ocenimo kombinatorično časovno zahtevnost algoritma 43.1. Ker sta algoritma ADJACENT in OPPOSITE kombinatorične časovne zahtevnosti $\mathcal{O}(1)$, je tudi algoritem 43.1 kombinatorične časovne zahtevnosti $\mathcal{O}(1)$.

Algoritem 43.1. Izračun začetnih točk dosegljivih intervalov diagrama.

Vhod: krivulji f, g, dosegljivi točki l_1, l_2 na levi stranici diagrama, dosegljivi točki b_1, b_2 na spodnji stranici diagrama, testna razdalja ε

Izhod: petorček (t_1, t_2, r_1, r_2, b), kjer sta t_1, t_2 vrednosti parametra funkcije f začetnih točk dosegljivih intervalov na zgornji stranici diagrama, r_1, r_2

vrednosti parametra funkcije g začetnih točk dosegljivih intervalov na desni stranici diagrama, b pa dostopnost zgornje desne točke diagrama

1: **procedure** REACHABLE($f, g, l_1, l_2, b_1, b_2, \varepsilon$)
2: $(t11, t12, b_1) \leftarrow$ ADJACENT(f, g, l_1, ε),
3: $(t21, t22, b_2) \leftarrow$ ADJACENT(f, g, l_2, ε)
4: $(t31, t32, b_3) \leftarrow$ OPPOSITE(g, f, b_1, ε),
5: $(t41, t42, b_4) \leftarrow$ OPPOSITE(g, f, b_2, ε)
6: $(r11, r12, b_5) \leftarrow$ ADJACENT(g, f, b_1, ε),
7: $(r21, r22, b_6) \leftarrow$ ADJACENT(g, f, b_2, ε)
8: $(r31, r32, b_7) \leftarrow$ OPPOSITE(f, g, l_1, ε),
9: $(r41, r42, b_8) \leftarrow$ OPPOSITE(f, g, l_2, ε)
10: $t_1 \leftarrow$ MIN($t11, t21, t31, t41$),
11: $t_2 \leftarrow$ MIN($t12, t22, t32, t4$))
12: $r_1 \leftarrow$ MIN($r11, r21, r31, r41$),
13: $r_2 \leftarrow$ MIN($r12, r22, r32, r42$)
14: $b = b_1 \vee b_2 \vee b_3 \vee b_4 \vee b_5 \vee b_6 \vee b_7 \vee b_8$
15: **return** (t_1, t_2, r_1, r_2, b)

Izrek 44.1. *Za podana zlepka $F = \{f_1, f_2, \ldots, f_n\}$ in $G = \{g_1, g_2, \ldots, g_m\}$ ravninskih gladkih krivulj, obstaja algoritem, ki za dano vrednost ε odloči, ali velja $\delta_F(F, G) \leq \varepsilon$, in ima kombinatorično časovno zahtevnost $\mathcal{O}(nm)$.*

Dokaz. Z uporabo do sedaj opisanih algoritmov, lahko odločitveni algoritem konstruiramo kot algoritem 44.1.

Ocenimo kombinatorično časovno zahtevnost algoritma 44.1. Klica algoritma PREPROCESS vrstic 2 in 3 sta časovne zahtevnosti $\mathcal{O}(n)$ in $\mathcal{O}(m)$, v tem vrstnem redu, množici krivulj F', G' pa sta le za konstantni faktor večji kot množici F, G, torej $\mathcal{O}(n)$, $\mathcal{O}(m)$, v tem vrstnem redu. Vrstica 10 se zato izvede $\mathcal{O}(nm)$–krat, njena časovna zahtevnost pa je $\mathcal{O}(1)$. Kombinatorična časovna zahtevnost algoritma 44.1 je torej dominirana s časovno zahtevnostjo zanke vrstice 8, torej $\mathcal{O}(nm)$.

Algoritem 44.1. Odločitveni problem za $\delta_F(F, G)$.

Vhod: zlepka krivulj $F = \{f_1, f_2, \ldots, f_n\}, G = \{g_1, g_2, \ldots, g_m\}$, testna razdalja ε

Izhod: odločitev, ali za množici krivulj F, G velja $\delta_F(F, G) \leq \varepsilon$

1: **procedure** DECISION(F, G, ε)
2: $\{f'_1, f'_2, \ldots, f'_{n'}\} = F' \leftarrow$ PREPROCESS(F, ε),
3: $\{g'_1, g'_2, \ldots, g'_{m'}\} = G' \leftarrow$ PREPROCESS(G, ε)
4: **if** $\|f'_1(\min(f'_1)) - g'_1(\min(g'_1))\| \leq \varepsilon$ **then**
5: $l1_{1,1} \leftarrow \min(g'_1), l2_{1,1} \leftarrow$ null
6: $b1_{1,1} \leftarrow \min(f'_1), b2_{1,1} \leftarrow$ null
7: **else return** false
8: **for** $i \leftarrow 1, 2, \ldots, n'$ **do**
9: **for** $j \leftarrow 1, 2, \ldots, m'$ **do**
10: $(b1_{i+1,j}, b2_{i+1,j}, l1_{i,j+1}, l2_{i,j+1}, b_{i,j}) \leftarrow$ REACHABLE($f'_i, g'_j, l1_{i,j}, l2_{i,j}, b1_{i,j}, b2_{i,j}$)
11: **return** $b_{n',m'}$

□

Poglavje 5

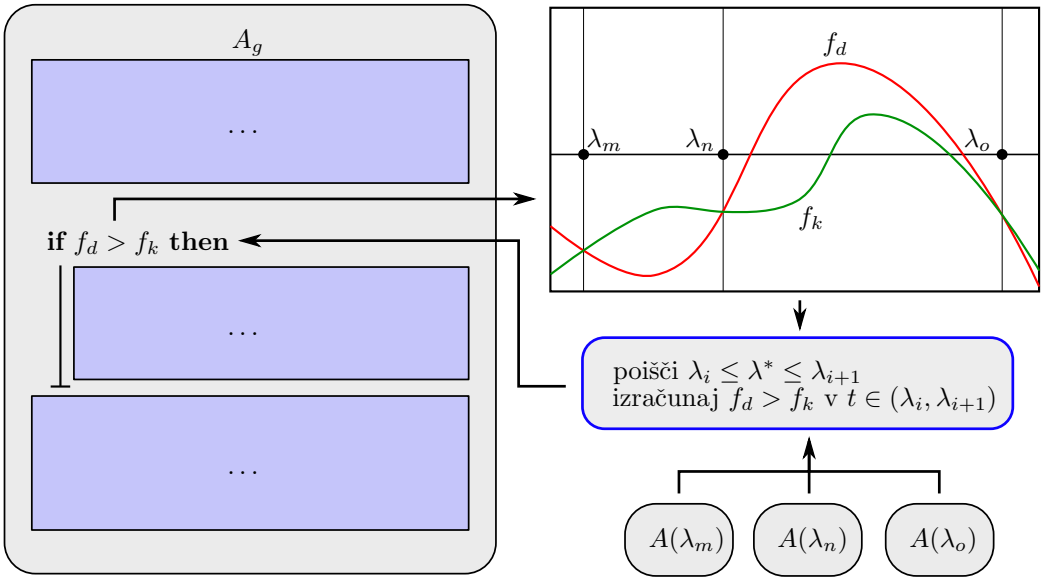
Rešitev optimizacijskega problema

Do sedaj smo konstruirali algoritem, ki rešuje odločitveni problem Fréchetove razdalje za zlepka gladkih ravninskih krivulj. V tem razdelku si bomo ogledali tehniko z imenom *parametrično iskanje*. Ogledali si bomo, kako parametrično iskanje deluje v splošnem, nato pa ga bomo uporabili v konstrukciji algoritma za iskanje optimizacijskega problema Fréchetove razdalje.

5.1 Parametrično iskanje

Naj bo $A(\lambda)$ algoritem, ki za problem odloči, ali velja $\lambda < \lambda^*$, $\lambda > \lambda^*$ ali $\lambda = \lambda^*$. Tok algoritma A naj bo odvisen od primerjav vrednosti $t_1, t_2, \ldots, t_n$. Parametrično iskanje uporabja generični algoritem A_g algoritma A na do tedaj še neznanem parametru λ^*. Pri tem primerjave problema A postanejo primerjave vrednosti polinomov $t_1(\lambda^*), t_2(\lambda^*), \ldots, t_n(\lambda^*)$.

Med izvajanjem algoritma A_g hranimo interval vrednosti $I = (l, h)$, znotraj katerega zagotovo leži λ^*. Na začetku je I kar interval $(-\infty, \infty)$. Ideja parametričnega iskanja je, da se interval I pri vsaki primerjavi dovolj zmanjša, da se polinoma dotične primerjave na območju I ne sekata, torej ju lahko primerjamo pri kateremkoli parametru $t \in I$.



Slika 48.1: Ilustracija parametričnega iskanja z uporabo generičnega algoritma A_g.

To se zgodi po postopku, ki ga prikazuje slika 48.1: kadarkoli primerjamo polinoma $t_i(\lambda), t_j(\lambda)$, izračunamo kritične vrednosti $\lambda_1, \lambda_2, \ldots, \lambda_K$, pri katerih se rezultat primerjave spremeni. Take kritične vrednosti so natanko koreni razlike polinomov $t_i(\lambda) - t_j(\lambda)$. Med kritičnimi vrednostmi lahko z uporabo algoritma A poiščemo dve zaporedni vrednosti $\lambda_k < \lambda^* < \lambda_{k+1}$. Interval I posodobimo kot $I \cap (\lambda_k, \lambda_{k+1})$, ker se polinoma t_i, t_j na intervalu $(\lambda_k, \lambda_{k+1})$ ne sekata, ju lahko primerjamo v katerikoli točki $t \in I$. Seveda lahko v primeru, ko algoritem A v katerem od korenov r_i vrne odgovor $\lambda = \lambda^*$, izvajanje prekinemo, in kot rezultat vrnemo točno vrednost λ^*.

Prikažimo delovanje parametričnega iskanja na primeru iz [18]. Naj bodo funkcije $f_i(\lambda) = a_i + b_i\lambda$ ter naj velja $b_i > 0$. Definirajmo funkcijo $F(\lambda)$ kot vrednosti mediane množice $\{f_1(\lambda), f_2(\lambda), \ldots, f_n(\lambda)\}$. Funkcija F je monotono naraščajoča. Iščemo enolično vrednost λ^*, da velja $F(\lambda^*) = 0$. Naj algoritem A rešuje problem $F(\lambda) \gtreqless 0$. Časovna zahtevnost algoritma A je $\mathcal{O}(n)$, saj mediano množice elementov lahko z algoritmom BFPRT poiščemo v linearnem času. Naj bo A_g generični algoritem, ki išče vrednost mediane polinomov

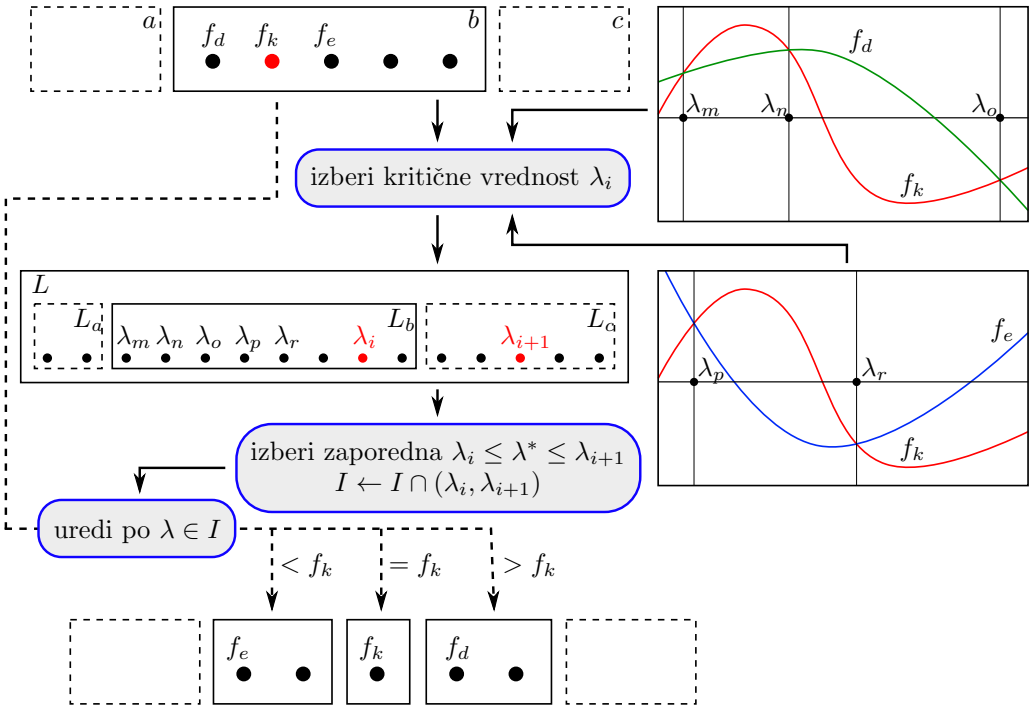
$\{f_1(\lambda^*), f_2(\lambda^*), \ldots, f_n(\lambda^*)\}$ pri vnaprej neznanem λ^*. Poglejmo postopek, ko algoritem primerja polinoma f_i, f_j. Primerjava je odvisna od kritične vrednosti λ_{ij}, ki je izražena po $a_i + b_i\lambda_{ij} = a_j + b_j\lambda_{ij}$. Potem se s klicem $A(\lambda_{ij})$ lahko odločimo, na kateri strani λ_{ij} leži λ^*, saj za $F(\lambda_{ij}) > 0$ velja $\lambda^* < \lambda_{ij}$, za $F(\lambda_{ij}) < 0$ pa velja $\lambda^* > \lambda_{ij}$. Glede na pozicijo λ^* lahko ustrezno posodobimo interval I. Ker polinom $f_i - f_j$ na intervalu I nima korenov, lahko polinoma f_i, f_j primerjamo v katerikoli točki $t \in I$. Ker za iskanje mediane potrebujemo $\mathcal{O}(n)$ primerjav, vsaka pa lahko zahteva klic algoritma A, je skupna časovna zahtevnost algoritema A_g, enaka $\mathcal{O}(n^2)$.

Če bi lahko več korenov λ_i različnih primerjav združili, bi lahko časovno zahtevnost algoritem še nekoliko zmanjšali. Naj bo A_p vzporedna izvedenka algoritma A_g, ki se izvede v N obhodih na P procesorjih. Ker vzporednost algoritma A_p zgolj simuliramo, lahko uporabimo šibek vzporedni model. V enem obhodu algoritma A_p se izvaja P primerjav polinomov vzporedno. Kritične vrednosti (korene razlik takih polinomov) lahko združimo v množico $L = \{\lambda_0, \lambda_1, \ldots, \lambda_{kP}\}$, $\lambda_0 \leq \lambda_1 \leq \cdots \leq \lambda_{kP}$. Ni potrebno, da je množica L urejena. Z uporabo algoritma za iskanje mediane poiščemo zaporedni kritični vrednosti, da velja $\lambda_i < \lambda^* < \lambda_{i+1}$. Pri tem uporabimo $kP + kP/2 + kP/4 + \cdots = \mathcal{O}(P)$ primerjav za iskanje median ter $\mathcal{O}(\log P)$ klicev algoritma A. Če je časovna zahtevnost algoritma A enaka $\mathcal{O}(T_A)$, je časovna zahtevnost parametričnega iskanja enaka $\mathcal{O}((P + T_A \cdot \log P) \cdot N)$. [18]

Ker je v praksi pogosto težko implementirati vzporedni algoritem A_p, lahko namesto njega uporabimo katerikoli vzporedni algoritem, ki izvede vsaj tiste primerjave polinomov, ki bi jih izvedel tudi algoritem A_g. Kot primer si oglejmo implementacijo vzporednega iskanja z uporabo algoritma za urejanje Quicksort [21].

Izrek 49.1. *Z uporabo algoritma za sortiranje* Quicksort *namesto vzporednega generičnega algoritma A_p, lahko za polinome $f_1, f_2, \ldots, f_n$ izračunamo interval I v pričakovanem času $\mathcal{O}((n + T_A \cdot \log n) \cdot \log n)$, kjer je $\mathcal{O}(T_A)$ časovna zahtevnost algoritma A.*

Dokaz. Grafični prikaz enega obhoda algoritma prikazuje slika 50.1. V obhodu



Slika 50.1: Ilustracija enega nivoja parametričnega iskanja z uporabo algoritma za urejanje QUICKSORT.

j za vsako množico k izberemo pivotni polinom f_k, na sliki je ta označen z rdečo barvo. Primerjave znotraj te množice obhoda j so primerjave med elementom f_k ter ostalimi elementi. Za vsako tako primerjavo polinomov f_i, f_j izračunamo množico kritičnih točk L_k ter vse množice združimo v množico L. Znotraj množice kritičnih točk L z uporabo algoritma za iskanje mediane ter odločitvenega algoritma A določi dve zaporedni točki λ_i, λ_{i+1}, med katerima leži optimalna vrednost λ^* ter posodobi interval I. Glede na novi interval I izračuna rezultat vseh primerjav ter glede na te primerjave razdeli množico na tri dele: polinome manjše od f_k, polinome enake f_k, ter polinome večje od f_k. Algoritem teče, dokler vsi polinomi niso urejeni.

Za n vhodnih polinomov je pričakovano število obhodov algoritma QUICKSORT $\mathcal{O}(\log n)$, v vsakem obhodu pa je potrebnih $\mathcal{O}(n)$ primerjav za iskanje median ter $\mathcal{O}(\log n)$ klicev algoritma A. Pričakovana časovna zahtevnost parametričnega iskanja z uporabo algoritma za urejanje QUICKSORT je torej

$\mathcal{O}((n + T_A \cdot \log n) \cdot \log n)$. □

Naj bo QuicksortParametricSearch(P, A) algoritem, ki nad množico polinomov P in odločitvenim algoritmom A izračuna interval I.

5.2 Implementacija

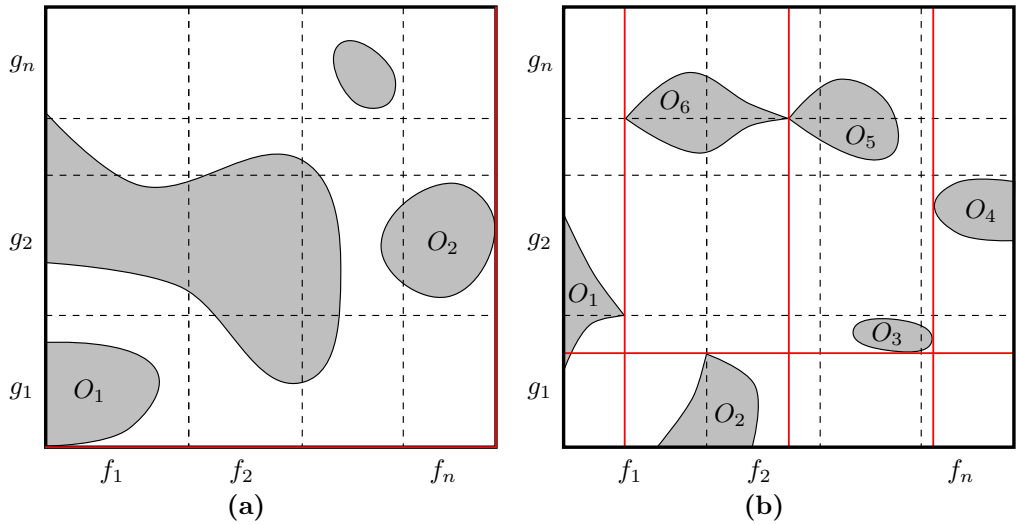
Za optimizacijski problem Fréchetove razdalje podajmo še nekaj dodatnih omejitev. Naj bo F zlepek gladkih algebraičnih krivulj $f_1, f_2, \ldots, f_n$ in G zlepek gladkih algebraičnih krivulj $g_1, g_2, \ldots, g_m$, katerih stopnja je omejena. S tem zagotovimo, da se v diagramu krivulj f_i, g_j pojavi le konstantno število ovir. Brez škode za splošnost lahko predpostavimo $f_i : [i-1, i] \to \mathbb{R}^2$, $g_i : [i-1, i] \to \mathbb{R}^2$. Koordinate mreže diagramov množic F, G naj bodo podane s parom (a, b), $a \in [0, n]$, $b \in [0, m]$.

Želimo izračunati natančno vrednost $\varepsilon^* = \delta_F(F, G)$. Začnemo z $\varepsilon = 0$ in postopoma povečujemo ε, dokler v mreži diagramov ne obstaja monotona pot iz točke $(0, 0)$ do točke (n, m). Razdelimo problem iskanja $\delta_F(F, G)$ na iskanje števila ε_i izmed množice *kritičnih vrednosti* $K = \{\varepsilon_1, \varepsilon_2, \ldots, \varepsilon_k\}$. Vrednost ε je kritična, ko se v neki diagramu krivulj f_i, g_j, $f_i \in F, g_j \in G$ pojavi eden od naslednjih primerov:

1. Odpre se nova vodoravna ali navpična pot na meji mreže diagramov, omejena z oviro. Tako situacijo prikazuje slika 52.1(a). Ločimo dva primera:
   - ovira seka stranico mreže diagramov tako, da presečišče leži v enem od kotov mreže (ovira O_1).
   - ekstremna točka ovire se dotika meje mreže diagramov (ovira O_2).
2. Odpre se nova vodoravna ali navpična pot omejena z dvema ovirama: to situacijo prikazuje slika 52.1(b). Ločimo tri različne primere:
   - ekstremni točki ovir ležita na isti premici (oviri O_3 in O_4).

- ekstremna točka ene ovire in presečišče druge ovire s stranico diagrama ležita na isti premici (oviri O_3 in O_2).
- presečišči ovir s stranicama diagrama ležita na na isti premici (oviri O_5 in O_6, ter oviri O_1 in O_6).

Ni nujno, da sta taki dve oviri znotraj istega diagrama, morata pa biti znotraj diagramov v isti vrstici (stolpcu) mreže, kot to prikazuje slika.



Slika 52.1: Možne situacije kritičnih točk.

Predpostavimo, da obstajajo naslednje primitivne operacije, ki so odvisne od opisa krivulj:

- ObstacleDiagram($f, g, s \in \{b, t, l, r\}$): za diagram algebraičnih krivulj f, g vrne množico vrednosti razdalj ε, kjer se ovire znotraj diagrama dotikajo stranice s, kot je to opisano v točki 1. Po predpostavki o tipu krivulj f, g je množica vrednosti omejena s konstanto.
- CriticalPoints(f, g): za diagram algebraični krivulji f, g vrne množico polinomov $\{p_1(\varepsilon), p_2(\varepsilon), \ldots, p_n(\varepsilon)\}$ koordinat ekstremnih točk ovir ter njihovih presečišč z mejo diagrama funkcij f, g. Po predpostavki o tipu krivulj f, g je množica polinomov omejena s konstanto.

Izrek 53.1. *Za dana zlepka F gladkih algebraičnih krivulj $f_1, f_2, \ldots, f_n$ in G gladkih algebraičnih krivulj $g_1, g_2, \ldots, g_m$, obstaja algoritem, ki izračuna vrednost $\delta_F(F, G)$, in ima pričakovano kombinatorično časovno zahtevnost $\mathcal{O}\big(nm \log^2 nm\big)$.*

Dokaz. Razdelimo algoritem na dva dela glede na tip kritičnih vrednosti:

1. Kritične vrednosti točke 1. Za vsak diagram na meji mreže diagramov z uporabo algoritma ObstacleDiagram določimo kritične vrednosti ε in jih združimo v množico $M = \{\varepsilon_1, \varepsilon_2, \ldots, \varepsilon_{k \cdot mn}\}$, $\varepsilon_1 < \varepsilon_2 < \cdots < \varepsilon_{k \cdot mn}$. Nato z uporabo odločitvenega algoritma Decision poiščemo dva zaporedna $\varepsilon_i < \varepsilon^* < \varepsilon_{i+1}$. Implementacijo algoritma predstavlja algoritem 53.1.

Algoritem 53.1. Prvi del optimizacijskega algoritma.

Vhod: zlepka algebraičnih krivulj $F = \{f_1, f_2, \ldots, f_n\}$, $G = \{g_1, g_2, \ldots, g_m\}$
Izhod: interval (l, h), da velja $l < \varepsilon^* < h$

```
1:  procedure OptimizationPreprocess1(F, G)
2:      M ← {−∞, ∞}
3:      for all f_i ∈ F do M ← M ∪ ObstacleDiagram(f_i, g_1, b)
4:      for all f_i ∈ F do M ← M ∪ ObstacleDiagram(f_i, g_m, t)
5:      for all g_i ∈ G do M ← M ∪ ObstacleDiagram(f_1, g_i, l)
6:      for all g_i ∈ G do M ← M ∪ ObstacleDiagram(f_n, g_i, r)
7:      M ← Sort(M)
8:      l ← 0, h ← |M|
9:      while h − l ≠ 1 do
10:         m ← (h + l)/2, d_m ← Decision(F, G, M_m)
11:         if d_m then h ← m
12:         else l ← m
13:     return (M_l, M_h)
```

Ocenimo časovno zahtevnost algoritma 53.1. Ker je velikost množice rezultatov algoritma ObstacleDiagram omejena s konstanto, je velikost množice M enaka $\mathcal{O}(n+m)$, kombinatorična časovna zahtevnost vrstic 3–6 pa $\mathcal{O}(n+m)$. Urejanje v vrstici 7 je časovne zahtevnosti $\mathcal{O}((n+m)\log(n+m))$. Zanka vrstice 9 ima $\mathcal{O}(\log(n+m))$ obhodov, klic algoritma Decision pa ima kombinatorično časovno zahtevnost $\mathcal{O}(nm)$. Skupna kombinatorična časovna zahtevnost algoritma 53.1 je torej $\mathcal{O}(nm\log(n+m))$.

2. Kritične vrednosti točke 2. Naj bo P unija množic polinomov, ki jo vrne algoritem CriticalPoints za vsak par f_i, g_j, $1 \leq i \leq n$, $1 \leq j \leq m$ zlepkov F, G. Množica P^F naj predstavlja vrednosti parametra zlepka F množice točk P, Množica P^G pa vrednosti parametra zlepka G množice točk P. Za kritično točko ε optimizacijskega problema mora veljati $P_i^F(\varepsilon) = P_j^F(\varepsilon)$ ali $P_i^G(\varepsilon) = P_j^G(\varepsilon)$ za nek par i, j. To pomeni, da za izračun vrednosti ε^* lahko uporabimo parametrično iskanje z uporabo algoritma za urejanje Quicksort, kjer najprej sortiramo vrednosti P_i^F, nato pa še vrednosti P_j^G.

Algoritem 54.1. Drugi del optimizacijskega algoritma.

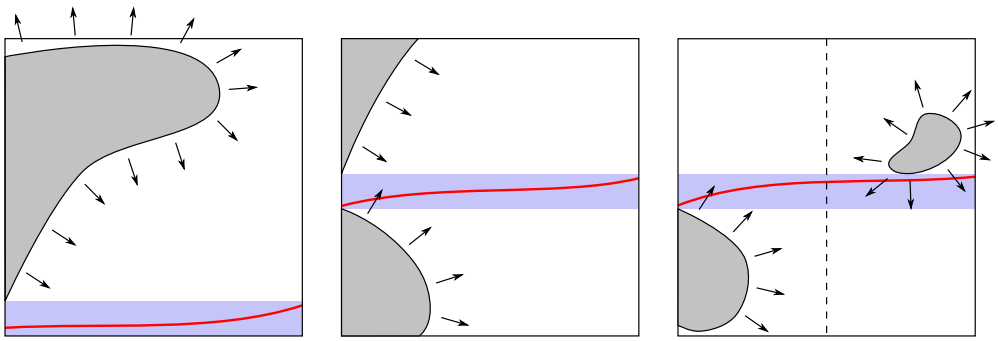
Vhod: zlepka algebraičnih krivulj $F = \{f_1, f_2, \ldots, f_n\}$, $G = \{g_1, g_2, \ldots, g_m\}$

Izhod: interval (l, h), da velja $l < \varepsilon^* < h$

```
1: procedure OptimizationPreprocess2(F, G)
2:     P ← ∅
3:     for all f_i ∈ F do
4:         for all g_j ∈ G do
5:             P ← P ∪ CriticalPoints(f_i, g_j)
6:     I ← QuicksortParametricSearch(P^F, Decision)
7:     J ← QuicksortParametricSearch(P^G, Decision)
8:     return I ∩ J
```

Analizirajmo časovno zahtevnost algoritma 56.1. Vrstica 2 po analizi iz prejšnje točke zahteva $\mathcal{O}(nm \log(n+m))$ časa. Ker je velikost množice rezultata algoritma CriticalPoints vrstice 3 konstantna, je velikost množice P enaka $\mathcal{O}(nm)$, pričakovana časovna zahtevnost vrstic 6,7 je torej $\mathcal{O}\big(nm \log^2 nm\big)$, kar je tudi pričakovana časovna zahtevnost algoritma.

Izrek 55.1. *Za vrednost $\delta = \delta_F(F, G)$ zlepek F krivulj $f_1, f_2, \ldots, f_n$ in zlepek G krivulj $g_1, g_2, \ldots, g_m$ se v nekem diagramu krivulj f_i, g_j pojavi ena od situacij kritične točke, opisanih v točkah 1, 2.*



Slika 55.1: Ilustracija izreka 55.1.

Dokaz. Privzemimo, da velja $\delta = \delta_F(F, G)$ ter se za noben par krivulj f_i, g_j ne pojavi situacija kritične točke. Potem v vsakem diagram krivulj f_i, g_j, skozi katero potuje monotona pot od točke $(0, 0)$ do točke (n, m) obstaja trak dostopnih točk, ki jo obdaja, kot to prikazuje slika 55.1. Ker za vsak $\varepsilon > 0$ velja $F^-_{\delta+\varepsilon}(f_i, g_j) \subset F^-_{\delta}(f_i, g_j)$ sledi, da obstaja nek ε, da za vrednost $\delta' = \delta - \varepsilon$ obstaja monotona pot iz točke $(0, 0)$ do točke (n, m), kar je v protislovju s predpostavko $\delta = \delta_F(F, G)$. □

Algoritma 53.1 in 54.1 lahko združimo v končni algoritem 56.1 za izračun $\delta_F(F, G)$. Ker je po izreku 55.1 točna vrednost ε^* zagotovo ena od kritičnih točk, je $\delta_F(F, G)$ kar zgornja meja preseka intervalov, ki ju vrneta klica algoritmov OptimizationPreprocess1 ter OptimizationPreprocess2.

Algoritem 56.1. Optimizacijski algoritem.

Vhod: zlepka algebraičnih krivulj $F = \{f_1, f_2, \ldots, f_n\}$, $G = \{g_1, g_2, \ldots, g_m\}$

Izhod: vrednost $\delta_F(F, G)$

1: **procedure** OPTIMIZATION(F, G)
2: $I \leftarrow$ OPTIMIZATIONPREPROCESS1(F, G)
3: $J \leftarrow$ OPTIMIZATIONPREPROCESS2(F, G)
4: **return** $\max(I \cap J)$

□

Poglavje 6

Rezultati

Na podlagi algoritmov iz poglavij 4 in 5 smo v programskem jeziku `C++` implementirali knjižnico, ki omogoča implementacijo razširitve odločitvenega in optimizacijskega problema Fréchetove razdalje zlepkov gladkih krivulj. Pri tem smo uporabili podatkovni tip za eksaktno aritmetiko `real` knjižnice `leda`[1].

Knjižnico sestavljata dva glavna razreda: razred `curve_segment` ter razred `free_space`.

6.1 Odločitveni problem

Pri razširitvi odločitvenega problema je potrebno implementirati razred krivulje, ki jo uporabljamo v zlepku, ter ga razširiti z metodami razreda `curve_segment`. Pri tem je potrebno implementirati metode, ki smo jih predstavili v razdelku 4.2.1.

Delovanje odločitvenega problema smo preizkusili na zlepkih daljic in krožnih odsekov. Ker je vzporedna krivulja daljici prav tako daljica, ter vzporedna krivulja krožnemu odseku prav tako krožni odsek, je zgornje funkcije preprosto implementirati.

Časovno zahtevnost odločitvenega problema smo v praksi preverili nad

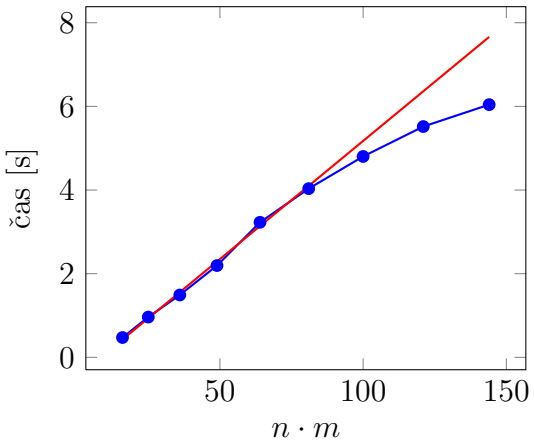
[1] `http://www.algorithmic-solutions.com/`

poligonalnimi krivuljami. Čase izvajanja prikazujeta tabela 58.1 in slika 58.2. Krivulje smo določili tako, da smo za vsako krivuljo naključno izbrali točke iz območja $[0,1]^2$. Za vsak par velikosti krivulj n, m smo generirali 1000 parov krivulj, ter čase izvajanja algoritma za odločitveni problem povprečili. Ker smo uporabili podatkovni tip za eksaktno aritmetiko, se pogosto dogaja, da primerjanje takih dveh vrednosti, ko sta ti dve enaki, traja veliko časa. To je tudi razlog za veliko standardno deviacijo meritev, kot je to prikazano v tabeli 58.1. Pri večjih vrednostih n, m se v mreži diagramov $D_\varepsilon^{F,G}$ zlepkov F, G pojavi precej diagramov, ki so v celoti bodisi znotraj $F_\varepsilon^-(F,G)$ bodisi znotraj $F_\varepsilon^+(F,G)$. Izračun algoritma REACHABLE je za tak diagram trivialen, zato se pri večjih vrednostih n, m krivulja časovne zahtevnosti nekoliko odkloni navzdol od linearne.

Časovne zahtevnosti zlepkov krožnih odsekov v praksi nismo uspeli preveriti, saj je čas izvajanja algoritma nad zlepki krožnih odsekov precej daljši od časa izvajanja nad poligonalnimi krivuljami, standardni odklon pa velik, zato nismo uspeli narediti dovolj meritev, da bi bili izsledki merodajni.

n,m	$n \cdot m$	čas [s]	σ [s]
4	16	0.47	1.07
5	25	0.96	1.88
6	36	1.49	2.85
7	49	2.2	3.44
8	64	3.23	4.22
9	81	4.03	5.29
10	100	4.8	5.77
11	121	5.52	7.17
12	144	6.04	7.24

Tabela 58.1: Časi izvajanja in standardna deviacija odločitvenega problema nad poligonalnimi krivuljami.



Slika 58.2: Grafična predstavitev tabele 58.1.

6.2 Optimizacijski problem

Pri razširitvi optimizacijskega problema je, podobno kot za odločitveni problem, potrebno implementirati razred krivulje, ki jo uporabljamo v zlepku, ter ga razširiti z metodami razreda `curve_segment`. Pri tem je potrebno poleg primitivnih operacij za odločitveni problem implementirati tudi primitivne operacije, predstavljene v razdelku 5.2.

Delovanje optimizacijskega problema smo preizkusili le nad poligonalnimi krivuljami, saj se je implementacija funkcije CriticalPoints za krožne odseke izkazala za praktično neizvedljivo.

Časovne zahtevnosti v praksi tudi tu zaradi pretirane porabe časa eksaktne aritmetike nismo uspeli preveriti.

Poglavje 7

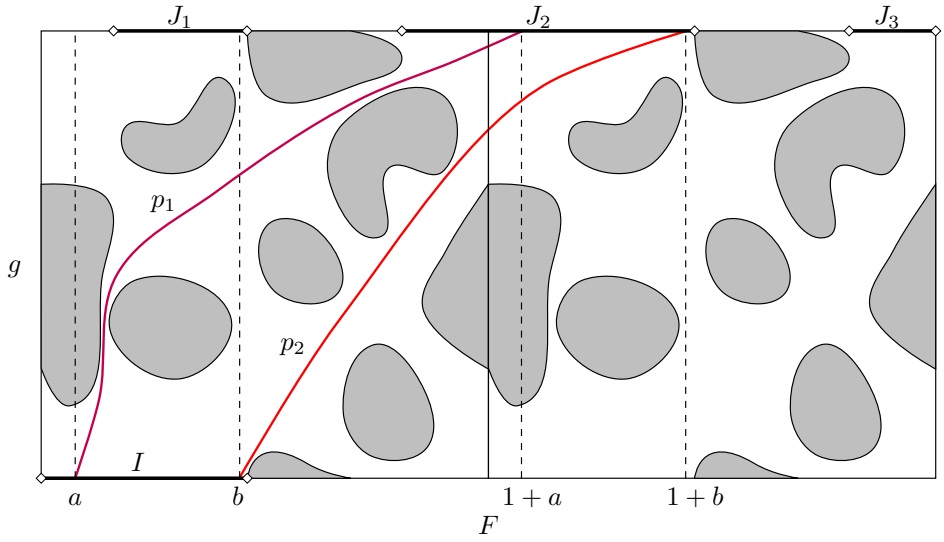
Zaključek

V nalogi smo predstavili algoritme in implementacijo za izračun odločitvenega problema Fréchetove razdalje gladkih krivulj in optimizacijskega problema Frechetove razdalje zlepkov gladkih algebraičnih krivulj. Implementacijo odločitvenega problema smo testirali nad zlepki daljic in krožnih odsekov, implementacijo optimizacijskega problema pa nad zlepki daljic. Sklepamo, da je implementacija z drugimi – bolj uporabnimi – družinami krivulj praktično neizvedljiva.

Zaradi uporabe eksaktne aritmetike časovne zahtevnosti nismo uspeli praktično preveriti, algoritmi pa so se izkazali za premalo numerično stabilne, da bi tipe za eksaktno aritmetiko lahko zamenjali s katerim od numeričnih tipov v plavajoči vejici fiksne dolžine.

V praksi je bolj smiselna uporaba diskretne Fréchetove razdalje, saj so v njej uporabljene primitivne operacije precej hitrejše, prav tako pa so v praksi krivulje večinoma podane diskretno. Ideja o Fréchetovi razdalji zlepkov gladkih krivulj je tako teoretično elegantna, vendar v praksi žal večinoma neuporabna.

Za konec si poglejmo še, kako bi lahko algoritme, predstavljene v nalogi, razširili na sklenjene zlepke gladkih krivulj. Brez škode za splošnost lahko predpostavimo, da sta krivulji definirani kot $f : [0, 1] \to \mathbb{R}^2$ in $g : [0, 1] \to \mathbb{R}^2$, in velja $f(0) = f(1)$, $g(0) = g(1)$. Ker sta krivulji sklenjeni, je smiselno



Slika 62.1: Ilustracija diagrama za sklenjeni krivulji f in g.

začetno/končno točko vsake krivulje izbrati poljubno.

Definirajmo zlepek $F : [0, 2] \to \mathbb{R}^2$ kot

$$F(t) = \begin{cases} f(t) & ; \quad 0 \leq t < 1 \\ f(t-1) & ; \quad 1 \leq t < 2 \end{cases}$$

Odločitveni problem Fréchetove razdalje lahko prevedemo na iskanje monotone poti v mreži diagrama $D^{F,g}$, ki se začne v točki $(a, 0)$ in konča v točki $(1 + a, 1)$, za nek $a \in [0, 1)$.

Za vsak interval dostopnih točk I na spodnji stranici mreže diagramov $D^{F,g}$ lahko določimo dosegljive intervale $J_1, J_2, \ldots, J_k$ na zgornji stranici, kot je to prikazano na sliki 62.1. Monotona pot, ki ustreza zgornjim kriterijem, obstaja, ko za nek interval J_i velja

$$\exists a \in [0, 1) : a \in I \wedge (a + 1) \in J_i. \tag{62.1}$$

Dve taki monotoni poti sta na sliki 62.1 označeni s p_1 in p_2.

Ko sta krivulji f in g zlepka gladkih krivulj, ki ustrezata omejitvam, podanim v poglavju 5, je število takih intervalov I in J_i omejeno s konstanto ter jih lahko izračunamo z uporabo algoritmov ADJACENT in OPPOSITE. Algoritem DECISION modificiramo tako, da rezultat izračuna iz enačbe (62.1). Kombinatorična časovna zahtevnost za izračun odločitvenega problema Fréchetove razdalje sklenjenih gladkih zlepkov velikosti n in m delov ostane enaka $\mathcal{O}(nm)$.

Za optimizacijski problem uporabimo algoritem OPTIMIZATION, v katerem uporabimo modificirani algoritem DECISION za sklenjene krivulje, kritične vrednosti pa izračunamo za mrežo diagramov $D^{F,g}$. Tudi tu kombinatorična časovna zahtevnost ostane enaka $\mathcal{O}\big(nm \log^2(nm)\big)$.

Literatura

[1] Pankaj K. Agarwal, Sariel Har-Peled, Nabil H. Mustafa, in Yusu Wang. Near-Linear Time Approximation Algorithms for Curve Simplification. Objavljeno v *Proceedings of the 10th Annual European Symposium on Algorithms*, ESA '02, str. 29–41. Springer-Verlag, 2002.

[2] Helmut Alt in Michael Godau. Computing The Fréchet Distance Between Two Polygonal Curves. *International Journal of Computational Geometry & Applications*, št. 5, zv. 1–2, str. 75–91, 1995.

[3] Helmut Alt, Christian Knauer, in Carola Wenk. Comparison of Distance Measures for Planar Curves. *Algorithmica*, št. 38, zv. 1, str. 45–58, 2003.

[4] Esther M. Arkin, L. Paul Chew, Daniel P. Huttnelocher, Klara Kedem, in Joseph S. B. Mitchell. An Efficient Computable Metric for Comparing Polygonal Shapes. *IEEE transactions on Pattern Analysis and Machine Intelligence*, št. 13, zv. 3, str. 209–216, 1991.

[5] Kevin Buchin, Maike Buchin, Christian Knauer, Günter Rote, in Carola Wenk. How difficult is it to walk the dog? Objavljeno v *Abstracts of the 23rd European Workshop on Computational Geometry*, str. 170–173, 2007.

[6] Kevin Buchin, Maike Buchin, Wouter Meulemans, in Bettina Speckmann. Locally Correct Fréchet Matchings. Objavljeno v Leah Epstein in Paolo Ferragina, uredniki, *Algorithms — ESA 2012*, zv. 7501 iz *Lecture Notes in Computer Science*, str. 229–240. Springer, 2012.

[7] Kevin Buchin, Maike Buchin, in André Schulz. Fréchet distance of surfaces: some simple hard cases. Objavljeno v *Proceedings of the 18th annual European conference on Algorithms: Part II*, ESA '10, str. 63–74. Springer-Verlag, 2010.

[8] Kevin Buchin, Maike Buchin, in Carola Wenk. Computing the Fréchet distance between simple polygons. *Computational Geometry, Theory and Applications*, št. 41, zv. 1–2, str. 2–20, 2008.

[9] Erin W. Chambers, Éric Colin de Verdière, Jeff Erickson, Sylvain Lazard, Francis Lazarus, in Shripad Thite. Homotopic Fréchet distance between curves or, walking your dog in the woods in polynomial time. *Computational Geometry, Theory and Applications*, št. 43, zv. 3, str. 295–311, 2010.

[10] Thomas Devogele. A New Merging Process for Data Integration Based on the Discrete Fréchet Distance. Objavljeno v *Advances in Spatial Data Handling*, str. 167–182. Springer, 2002.

[11] Anne Driemel, Sariel Har-Peled, in Carola Wenk. Approximating the Fréchet distance for realistic curves in near linear time. Objavljeno v *Proceedings of the 2010 annual symposium on Computational geometry*, SoCG '10, str. 365–374. ACM, 2010.

[12] Adrian Dumitrescu in Günter Rote. On the Fréchet distance of a set of curves. Objavljeno v *Proceedings of the 16th Canadian Conference on Computational Geometry*, CCCG '04, str. 162–165, 2004.

[13] Alon Efrat, Quanfu Fan, in Suresh Venkatasubramanian. Curve Matching, Time Warping, and Light Fields: New Algorithms for Computing Similarity between Curves. *Journal of Mathematical Imaging and Vision*, št. 27, zv. 3, str. 203–216, 2007.

[14] Thomas Eiter in Heikki Mannila. Computing Discrete Fréchet Distance. Technical Report CD-TR 94/64, Christian Doppler Laboratory for Expert Systems, 1994.

[15] Piotr Indyk. Approximate nearest neighbor algorithms for Fréchet distance via product metrics. Objavljeno v *Proceedings of the eighteenth annual symposium on Computational geometry*, SCG '02, str. 102–106. ACM, 2002.

[16] Atlas F. Cook IV in Carola Wenk. Geodesic Fréchet distance inside a simple polygon. *ACM Transactions on Algorithms*, št. 7, zv. 1, str. 9:1–9:19, 2010.

[17] Anil Maheshwari in Jiehua Yi. On Computing Fréchet Distance of Two Paths on a Convex Polyhedron. Objavljeno v *Proceedings of the 21st European Workshop on Computational Geometry*, str. 41–44, 2005.

[18] Nimrod Megiddo. Applying Parallel Computation Algorithms in the Design of Serial Algorithms. *Journal of the ACM*, št. 30, zv. 4, str. 852–865, 1983.

[19] Günter Rote. Computing the Fréchet distance between piecewise smooth curves. *Computational Geometry, Theory and Applications*, št. 37, zv. 3, str. 162–174, 2007.

[20] R. Sriraghavendra, Karthik K., in Chiranjib Bhattacharyya. Fréchet Distance Based Approach for Searching Online Handwritten Documents. Objavljeno v *Proceedings of the Ninth International Conference on Document Analysis and Recognition*, ICDAR '07, str. 461–465. IEEE Computer Society, 2007.

[21] René van Oostrum in Remco C. Veltkamp. Parametric Search Made Practical. *Computational Geometry, Theory and Applications*, št. 28, zv. 2–3, str. 75–88, 2004.