MATEMATICKO-FYZIKÁLNA FAKULTA
UNIVERZITY KOMENSKÉHO
v Bratislave

D I P L O M O V Á P R Á C A

Bratislava 2000 **Radovan Mikuš**

Matematicko-fyzikálna fakulta
Univerzity Komenského v Bratislave
Katedra geometrie

Rozmiestňovanie geometrických útvarov v diskrétnej rovine

Diplomová práca

Autor:
Radovan Mikuš

Vedúci diplomovej práce:
Doc. RNDr. Miloš Božek, CSc.

Čestne prehlasujem, že predkladanú prácu som vypracoval samostatne len s použitím literatúry, ktorá je uvedená v zozname.

V Bratislave, dňa 28. apríla 2000

. . . . . .

Radovan Mikuš

Touto cestou ďakujem vedúcemu diplomovej práce Doc. RNDr. Milošovi Božekovi, CSc. za cenné odborné rady, pripomienky a návrhy, ktoré mi pomohli pri realizácii výslednej podoby tejto práce.

Obsah

Úvod

Rozmiestňovanie geometrických útvarov v rovinnej oblasti má široké uplatnenie v mnohých odvetviach spracovateľského priemyslu (v textilnom, kožiarskom, obuvníckom, a pod.). Konkrétnym príkladom môže byť napríklad využitie rozmiestňovania útvarov v textilnom priemysle, kde je potrebné vystrihnúť jednotlivé dielce z kusu látky tak, aby sa minimalizoval odpad. Geometrickými útvarmi sú v tomto prípade dielce a rovinnou oblasťou je kus látky prípadne iného materiálu. Cieľom je vytvoriť schému rozmiestnenia dielcov, tzv. marker, podľa ktorého môžu postupovať automatické strihacie alebo rezacie zariadenia. Na rozmiestnenie zvyčajne bývajú kladené určité požiadavky. Prirodzenou požiadavkou kladenou na strihací plán je, aby sa dielce navzájom neprekrývali, ani nepresahovali cez hranicu materiálu. Snahou je rozmiestniť tieto dielce tak, aby po ich vystrihnutí bol odpad materiálu čo najmenší. Pre danú množinu útvarov je nájdenie optimálneho (z hľadiska využitia materiálu) strihacieho plánu *prakticky neriešiteľným problémom* (viď [26], [29]).

V princípe prevládajú dva prístupy riešenia tohto problému: *interaktívny* a *automatizovaný*. Pri interaktívnom prístupe sa pri hľadaní riešenia využívajú skúsenosti a intuícia človeka. Použitím CAD systému môžu trénovaní operátori tvoriť skoro optimálne markre manuálne, ale je to zložitá a časovo náročná úloha.

Podstatou automatizovaného prístupu je, že počítač rieši úlohu na základe vstupných údajov sám, bez pomoci operátora. V našom prípade je cieľom nájsť rozmiestnenie navzájom neprekrývajúcich sa útvarov v rovine, pričom toto rozmiestnenie musí navyše spĺňať určité kritéria. Jedným z kritérii môže byť požiadavka na maximálnu efektívnosť využitia oblasti geometrickými útvarmi, čo má v praxi za následok minimálny odpad materiálu. Pod efektívnosťou využitia oblasti geometrickými útvarmi budeme v tejto práci rozumieť pomer súčtu obsahov rozmiestnených útvarov ku obsahu ich obdĺžnikovej obálky.

Nájsť rozmiestnenie s maximálnou efektívnosťou využitia oblasti pre zadanú množinu útvarov je veľmi ťažké a časovo náročné. Všeobecný problém rozmiestnenia (sú povolené aj rotácie útvarov) je NP-ťažký, teda, za predpokladu, že $P \neq NP$, prakticky neriešiteľný. O translačnom probléme je dokázané, že je NP-úplný (viď [29]).

Pri automatizovanom prístupe je snahou vyvinúť taký algoritmus, ktorý by bol schopný nájsť riešenie v kratšom čase, ale s porovnateľnou alebo prípadne väčšou efektivitou ako človekom vytvorené rozmiestnenie. Je niekoľko možností ako pristupovať k riešeniu daného problému. Môžeme ich rozdeliť na tri hlavné prístupy: *náhodný* (alebo tiež stochastický), *systematický* a prípadne *kombinovaný*.

Najjednouchším riešením využívajúcim náhodný prístup je nasledujúce. Počítač náhodne

rozmiestni geometrické útvary v oblasti a porovná efektivitu rozmiestnenia s doteraz najlepšou nájdenou. Ak je efektivita menšia, algoritmus opäť vygeneruje náhodné rozmiestnenie a porovná ho. Tento proces opakuje vopred neurčený počet krát. Je zrejmé, že ani vykonanie veľkého množstva pokusov nemusí viesť k nájdeniu optimálneho riešenia. Pravdepodobnosť nájdenia nového, efektívnejšieho rozmiestnenia klesá s množstvom vykonaných pokusov.

Pri systematickom prístupe už nejde o náhodné pokusy ako v predchádzajúcom prípade, ale postupne sa generujú rozmiestnenia pomocou nejakého systému alebo stratégie. Avšak vykonávanie takéhoto algoritmu je priestorovo, ale najmä časovo veľmi náročné. Od vhodnosti zvolenej stratégie v podstatnej miere závisí výsledné riešenie. Zoberme si príklad využitia rozmiestňovania útvarov v textilnom priemysle. Empiricky sa zistilo, že najväčšie kusy, teda útvary s najväčším obsahom, by sa mali zaradiť do rozmiestnenia ako prvé (viď [29]). Až potom, keď nájdeme efektívne rozmiestnenie týchto útvarov, v našom prípade častí nohavíc, môžeme pristúpiť k rozmiestňovaniu menších častí ako vreciek, opaskov a pod. Inou možnosťou je rozdeliť oblasť na podmnožiny a riešiť úlohu zvlášť pre každú z nich. Použitá stratégia ale nemusí byť vyhovujúca pre iné aplikácie. Preto nájdenie všeobecnej stratégie použiteľnej pre ľubovoľnú úlohu je dodnes veľkým problémom.

V oboch uvedených prístupoch sa aplikujú, v závislosti od konkrétnej aplikácie, rôzne dodatočné podmienky dané vlastnosťami materiálov a obrábacích strojov. V prípade spracovania textílií sú napríklad prípustné nanajvýš rotácie o násobok 90° alebo do 1°, prípadne sú úplne zakázané. Geometrické útvary sa vo väčšine prípadov reprezentujú ako mnohouholníky a rovinná oblasť je nahradená obdĺžnikom s pohyblivou hranicou.

Doteraz publikované riešenia automatického rozmiestňovania útvarov pracovali zvyčajne s mnohouholníkovými aproximáciami dielcov látky, ktoré sa snažili rozmiestniť do oblasti, väčšinou obdĺžnikového tvaru, v euklidovskej rovine. V tejto práci narábame s iným typom aproximácie dielcov, a to s útvarmi definovanými pomocou bitových máp, ktoré rozmiestňujeme v diskrétnej rovine, ktorú si možno predstaviť ako nekonečnú mriežku bodov, kde každý bod má štyroch, ôsmych, prípadne šiestich susedov. Nespornou výhodou diskrétnej roviny je najmä to, že v nej možno efektívne implementovať viaceré algoritmy, ktorých obdoby pre euklidovskú rovinu je buď možné veľmi ťažko si predstaviť alebo nie sú implementovateľné s prakticky použiteľnou časovou zložitosťou.

Cieľ práce

Cieľom tejto práce je vytvoriť pevnú pôdu pre budúce práce v oblasti rozmiestňovania geometrických útvarov v diskrétnej rovine. Budú uvedené niektoré algoritmy, ktoré môžu viesť najmä ku zrýchleniu automatizovaného rozmiestňovania geometrických útvarov, ktoré sú reprezentované maticami pozostávajúcimi z núl a jednotiek.

Pri automatizovanom rozmiestňovaní sa často vyskytuje problém stanovenia, či dané dva geometrické útvary majú pre konkrétne vzájomné posunutie prienik alebo nie. Tento problém je možné transformovať pomocou Minkowského súčtov na úlohu zistiť, či daný bod patrí nejakému útvaru. Výpočet Minkowského súčtu dvoch mnohouholníkov v euklidovskej rovine nemá príliš veľkú časovú zložitosť, ale v diskrétnej rovine, kde geometrické útvary môžu byť skutočne ľubovoľného tvaru, je tento problém náročnejší. Ak by sme totiž mali dva útvary veľkosti rádovo

$n \times n$ bodov, výpočet Minkowského súčtu klasickým algoritmom by trval čas $\Theta(n^4)$, čo je pre väčšie hodnoty n prakticky neriešiteľný problém (v rámci niekoľkých hodín aj na rýchlejšom počítači), najmä vtedy, keď chceme spočítať Minkowského súčet pre každú dvojicu z nejakej množiny útvarov. V predkladanej práci bude prezentovaný algoritmus počítajúci Minkowského súčet dvoch maticovo reprezentovaných útvarov v asymptoticky lepšom čase $O(n^2 \lg n)$.

Ďalej bude uvedený algoritmus založený na efektívnom algoritme pre výpočet dvojrozmernej diskrétnej cyklickej konvolúcie, ktorý pre dané dva geometrické útvary zostaví tabuľku obsahujúcu pre každé možné posunutie, pri ktorom sa prekrývajú obálky útvarov, obsahy prienikov týchto útvarov pri tomto posunutí. Časová zložitosť tohto algoritmu je rovnaká ako u predchádzajúceho, t.j. $O(n^2 \lg n)$. Tento algoritmus má praktické využitie najmä pri optimalizačných problémoch rozmiestňovania útvarov, kde sa snažíme minimalizovať plochu prekryvu rozmiestňovaných útvarov, nehovoriac už o tom, že je na ňom postavený aj algoritmus na výpočet Minkowského súčtov.

Táto práca rozoberie viacero možností efektívneho výpočtu dvojrozmernej diskrétnej cyklickej konvolúcie – klasický algoritmus používajúci rýchlu Fourierovu transformáciu, nový algoritmus založený na diskrétnej cas-cas transformácii, ktorá má viacero výhod oproti Fourierovej transformácii, a napokon bude spomenutá možnosť vytvorenia algoritmu využívajúceho Fourierove transformácie v konečných poliach, ktorý by mal odstrániť niektoré nedostatky predchádzajúcich algoritmov.

Cieľom tejto práce nie je vytvoriť kompletné riešenie automatického rozmiestňovania geometrických útvarov, ale načrtnúť nový smer a vybudovať aparát pre rozmiestňovanie útvarov v diskrétnej rovine. Vzhľadom na veľkú zložitosť problému automatického rozmiestňovania geometrických útvarov nie je prakticky možné systematicky testovať všetky konfigurácie útvarov, preto sa ako alternatívne riešenie používajú rôzne približné a stochastické algoritmy. Niektoré z týchto metód bližšie popíšeme v kapitole 4. Ak pomocou nich chceme riešiť problém automatického rozmiestňovania geometrických útvarov, musíme popísať postup, ktorým možno navzájom porovnávať konfigurácie geometrických útvarov. Preto sa najprv zameriame na jednu z možných definícií ohodnotenia translačných konfigurácií, ktorej maximum budeme za pomoci týchto algoritmov hľadať. Prezentovaná metóda prináša niekoľko výhod, napríklad má také vlastnosti, ktoré umožňujú rýchlejšiu konvergenciu optimalizačných algoritmov.

Kapitola 1

Diskrétne ortogonálne transformácie

V tejto kapitole uvedieme definície niektorých diskrétnych transformácií, ich niektorých vlastností a budeme sa zaoberať niektorými metódami efektívneho výpočtu týchto transformácií. Cieľom tejto kapitoly je najmä uviesť čitateľa do problematiky diskrétnych transformácií a definovať notáciu, ktorú budeme neskôr používať. Azda najzaujímavejšou časťou tejto kapitoly môže byť odvodenie rýchleho algoritmu na výpočet diskrétnej Hartleyho transformácie.

1.1. Diskrétna Fourierova transformácia

V literatúre sa nachádzajú rôzne verzie definícií diskrétnej Fourierovej transformácie (skrátene DFT), ktoré sa líšia zvyčajne iba o konštantný faktor (viď napr. [7] vs. [33]). My budeme používať trochu inú definíciu, ktorá je výhodná najmä pri dokazovaní rôznych vlastností DFT.

Definícia 1.1 (Diskrétna Fourierova transformácia).
Nech $\mathbf{x}$ je vektor (postupnosť) komplexných čísel dĺžky N. *Jednorozmerná diskrétna Fourierova transformácia* vektora $\mathbf{x}$ je vektor $\Phi(\mathbf{x})$ dĺžky N, ktorého zložky sú definované vzťahom:

$$\Phi(\mathbf{x})_k = \frac{1}{\sqrt{N}} \sum_{n=0}^{N-1} \mathbf{x}_n \mathrm{e}^{\frac{-\mathrm{i}2\pi kn}{N}} , \quad k \in \{0,\ldots,N-1\} .$$

$\Phi(\mathbf{x})_k$ sú zvyčajne nazývané ako „Fourierove koeficienty“.

Vektor $\mathbf{x}$ možno vypočítať z $\Phi(\mathbf{x})$ použitím *inverznej* diskrétnej Fourierovej transformácie:

$$\mathbf{x}_n = \Phi^{-1}(\Phi(\mathbf{x}))_n = \frac{1}{\sqrt{N}} \sum_{k=0}^{N-1} \Phi(\mathbf{x})_k \mathrm{e}^{\frac{\mathrm{i}2\pi kn}{N}} , \quad n \in \{0,\ldots,N-1\} .$$

Lema 1.2 (O komplexnej združenosti DFT).
Inverzná diskrétna Fourierova transformácia komplexného vektora $\mathbf{x}$ je v nasledujúcom vzťahu s diskrétnou Fourierovou transformáciou[1]:

$$\Phi^{-1}(\mathbf{x}) = (\Phi(\mathbf{x}^*))^* .$$

[1] Výraz $\mathbf{x}^*$ označuje vektor (maticu), ktorého zložkami sú komplexne združené zložky vektora (matice) $\mathbf{x}$.

Dôkaz. Je relatívne jednoduchý a možno ho nájsť (v pozmenenom tvare, vzhľadom na rozdielnu definíciu DFT) napr. v [7, s. 124]. ■

Definícia 1.3 (Dvojrozmerná diskrétna Fourierova transformácia).
Nech $\mathbf{x}$ je matica komplexných čísel rozmerov $N_1 \times N_2$. *Dvojrozmerná diskrétna Fourierova transformácia* matice $\mathbf{x}$ je matica $\Phi_{2D}(\mathbf{x})$ rozmerov $N_1 \times N_2$, ktorej prvky sú definované:

$$\Phi_{2D}(\mathbf{x})_{k_1,k_2} = \frac{1}{\sqrt{N_1 N_2}} \sum_{n_1=0}^{N_1-1} \sum_{n_2=0}^{N_2-1} \mathbf{x}_{n_1,n_2} \mathrm{e}^{\frac{-\mathrm{i}2\pi k_1 n_1}{N_1}} \mathrm{e}^{\frac{-\mathrm{i}2\pi k_1 n_2}{N_2}} ,$$

kde $k_1 \in \{0,\ldots,N_1-1\}$ a $k_2 \in \{0,\ldots,N_2-1\}$. Podobne ako v jednorozmernom prípade, inverzná dvojrozmerná diskrétna Fourierova transformácia je definovaná:

$$\mathbf{x}_{n_1,n_2} = \Phi_{2D}^{-1}(\Phi_{2D}(\mathbf{x}))_{n_1,n_2} = \frac{1}{\sqrt{N_1 N_2}} \sum_{k_1=0}^{N_1-1} \sum_{k_2=0}^{N_2-1} \Phi_{2D}(\mathbf{x})_{k_1,k_2} \mathrm{e}^{\frac{\mathrm{i}2\pi k_1 n_1}{N_1}} \mathrm{e}^{\frac{\mathrm{i}2\pi k_2 n_2}{N_2}} ,$$

kde $n_1 \in \{0,\ldots,N_1-1\}$ a $n_2 \in \{0,\ldots,N_2-1\}$.

Lema 1.4 (O komplexnej združenosti dvojrozmernej DFT).
Inverzná dvojrozmerná diskrétna Fourierova transformácia matice $\mathbf{x}$ je v nasledujúcom vzťahu s dvojrozmernou DFT:

$$\Phi_{2D}^{-1}(\mathbf{x}) = \left(\Phi_{2D}(\mathbf{x}^*)\right)^* .$$

Dôkaz. Podobne ako dôkaz lemy 1.2, aj tento je jednoduchý, a preto sa ním bližšie nebudeme zaoberať. ■

1.2. Diskrétna Hartleyho transformácia

Diskrétna Hartleyho transformácia (ďalej DHYT) sa využíva predovšetkým pri spracovaní zvukových dát. Jej nespornou výhodou oproti diskrétnej Fourierovej transformácii je, že počas výpočtu DHYT postupnosti reálnych čísel narábame iba s reálnymi číslami[2].

Definícia 1.5 (Diskrétna Hartleyho transformácia).
Nech $\mathbf{x}$ je vektor reálnych čísel dĺžky N. *Diskrétna Hartleyho transformácia* vektora $\mathbf{x}$ je vektor $\overline{h}(\mathbf{x})$ dĺžky N, ktorého zložky sú definované vzťahom[3]

$$\overline{h}(\mathbf{x})_k = \frac{1}{\sqrt{N}} \sum_{n=0}^{N-1} \mathbf{x}_n \operatorname{cas}\left(\tfrac{2\pi kn}{N}\right) , \quad k \in \{0,\ldots,N-1\} ,$$

[2]Dalo by sa oponovať, že aj diskrétnu Fourierovu transformáciu možno vypočítať použitím iba reálnej aritmetiky, ak každé komplexné číslo reprezentujeme ako dvojicu reálnych čísel, avšak pri DHYT musíme pracovať iba s polovičným množstvom reálnych premenných.

[3]Podobnú definíciu (líšiacu sa o konštantný faktor) možno nájsť v [33, s. 24], [34, s. 18].

kde $\mathrm{cas}(\alpha) = \cos(\alpha) + \sin(\alpha)$. *Inverzná diskrétna Hartleyho transformácia* je daná rovnakým vzťahom[4]:

$$\mathbf{x}_n = \overline{h}^{-1}(\overline{h}(\mathbf{x}))_n = \frac{1}{\sqrt{N}} \sum_{k=0}^{N-1} \overline{h}(\mathbf{x})_k \,\mathrm{cas}\left(\tfrac{2\pi kn}{N}\right) ,$$

ktorý možno dokázať na základe rovnosti

$$\sum_{k=0}^{N-1} \mathrm{cas}\left(\tfrac{2\pi km}{N}\right) \mathrm{cas}\left(\tfrac{2\pi kn}{N}\right) = \begin{cases} N, & \text{ak } m = n\,, \\ 0, & \text{inak,} \end{cases}$$

za predpokladu, že m a n sú celé čísla z intervalu 0 až $N-1$.

Lema 1.6.
DHYT sa dá ľahko získať z DFT a naopak nasledujúcim spôsobom [33, s. 25]:

$$\begin{aligned} \overline{h}(\mathbf{x})_k &= \mathrm{Re}\{\Phi(\mathbf{x})_k\} - \mathrm{Im}\{\Phi(\mathbf{x})_k\} , \\ \mathrm{Re}\{\Phi(\mathbf{x})_k\} &= \frac{1}{2}\left(\overline{h}(\mathbf{x})_{N-k} + \overline{h}(\mathbf{x})_k\right) , \\ \mathrm{Im}\{\Phi(\mathbf{x})_k\} &= \frac{1}{2}\left(\overline{h}(\mathbf{x})_{N-k} - \overline{h}(\mathbf{x})_k\right) . \end{aligned}$$

Poznámka.
Dvojrozmerná DHYT nemá separovateľné jadro[5], a preto sa ňou nebudeme zaoberať. Transformácia so separovateľným Hartleyho transformačným jadrom sa nazýva *diskrétna cas-cas transformácia*.

1.2.1. Rýchly algoritmus výpočtu DHYT

V tomto odseku ukážeme, že existuje algoritmus počítajúci jednorozmernú diskrétnu Hartleyho transformáciu N-prvkovej postupnosti v čase $O(N \lg N)$.

Ak by sme vyčísľovali hodnoty výslednej postupnosti navzájom oddelene, t.j. podľa definície, počet násobení a sčítaní potrebných na výpočet jednorozmernej DHYT by bol rádovo N^2 (pretože treba vyhodnotiť N bodov, z ktorých každý vyžaduje použitie rádovo N aritmetických operácií). Teda čas výpočtu takéhoto algoritmu by zrejme bol $\Theta(N^2)$. Preto musíme zvoliť iný prístup. Ak by sa DHYT nedala počítať rýchlejšie, jej použiteľnosť by bola prakticky zanedbateľná.

Jedným zo spôsobov, ako možno vypočítať DHYT reálnej postupnosti $\mathbf{x}$ dĺžky N, je vypočítať najprv DFT tejto postupnosti a následne využiť lemu 1.6. Je to celkom rozumný spôsob, keďže existuje veľké množstvo rôznych algoritmov počítajúcich DFT veľmi efektívne pre prakticky ľubovoľné dĺžky vstupnej postupnosti. Napríklad populárne *Radix 2* algoritmy možno použiť, ak N je celočíselná mocnina 2. Tieto algoritmy majú časovú zložitosť iba $O(N \lg N)$, nakoľko

[4] Táto vlastnosť je výhodná z hľadiska jednoduchosti implementácie algoritmov využívajúcich doprednú aj inverznú DHYT.

[5] T.j., že neexistujú také dve funkcie f a g, pre ktoré by platila rovnosť $\mathrm{cas}(x+y) = f(x)g(x)$.

využívajú rekurzívnu dekompozíciu N-bodovej transformácie na dve $N/2$-bodové transformácie. Tento postup môže byť v princípe aplikovaný na akékoľvek zložené číslo N, jedinou nevýhodou je však väčšia náročnosť jeho implementácie. Tzv. *Radix 4* algoritmy pracujú na podobnom princípe ako Radix 2 algoritmy a zvyčajne sú o niečo rýchlejšie. Taktiež existuje efektívny algoritmus pre vyhodnocovanie DFT postupnosti prvočíselnej dĺžky založený na poznatkoch z teórie čísel (viď napr. [13]).

Ďalším možným spôsobom výpočtu DHYT je odvodiť rýchly *Radix 2* (FHYT) algoritmus pre túto transformáciu. Postup je podobný ako pri odvodzovaní algoritmu FFT, avšak je mierne komplikovanejší. Algoritmy FHYT sú v literatúre známe, ale ich bližší popis je veľmi ťažko dostupný. Preto ukážeme jeden zo spôsobov ako počítať diskrétnu Hartleyho transformáciu efektívnejšie, bez použitia dedikovaného algoritmu FFT.

Označme teraz $\mathrm{FHYT}(N,\mathbf{x})$ funkciu, ktorá vezme ako argumenty prirodzené číslo N a N-prvkovú postupnosť reálnych čísel $\mathbf{x}$ a vráti postupnosť $\sqrt{N}\,\overline{h}(\mathbf{x})$, ktorá má taktiež N prvkov, t.j.:

$$\mathrm{FHYT}(N,\mathbf{x})_k = \sqrt{N}\,\overline{h}(\mathbf{x})_k = \sum_{n=0}^{N-1} \mathbf{x}_n \operatorname{cas}\left(\tfrac{2\pi kn}{N}\right) .$$

Zaveďme dočasné označenie $\mathrm{ri}(x)$ pre výraz $\mathrm{Re}\{x\}+\mathrm{Im}\{x\}$. Takto prejde $\mathrm{FHYT}(N,\mathbf{x})_k$ do tvaru:

$$\mathrm{FHYT}(N,\mathbf{x})_k = \sum_{n=0}^{N-1} \mathbf{x}_n \,\mathrm{ri}\left(\mathrm{e}^{\frac{\mathrm{i}2\pi kn}{N}}\right) = \mathrm{ri}\left(\sum_{n=0}^{N-1} \mathbf{x}_n \mathrm{e}^{\frac{\mathrm{i}2\pi kn}{N}}\right) .$$

Ak je N párne, túto sumu možno rozdeliť na „párnu“ ($n=2n'$) a „nepárnu“ ($n=2n'+1$) časť, kde $n' \in \{0,\ldots,N/2-1\}$. Ak naviac položíme $\mathbf{y}_{n'} = \mathbf{x}_{2n'}$ a $\mathbf{z}_{n'} = \mathbf{x}_{2n'+1}$, môžeme písať:

$$\begin{aligned}
\mathrm{FHYT}(N,\mathbf{x})_k &= \mathrm{ri}\left(\sum_{n'=0}^{N/2-1} \mathbf{x}_{2n'} \mathrm{e}^{\frac{\mathrm{i}2\pi k(2n')}{N}} + \sum_{n'=0}^{N/2-1} \mathbf{x}_{2n'+1} \mathrm{e}^{\frac{\mathrm{i}2\pi k(2n'+1)}{N}}\right) = \\
&= \mathrm{ri}\left(\sum_{n'=0}^{N/2-1} \mathbf{y}_{n'} \mathrm{e}^{\frac{\mathrm{i}2\pi k(2n')}{N}}\right) + \mathrm{ri}\left(\sum_{n'=0}^{N/2-1} \mathbf{z}_{n'} \mathrm{e}^{\frac{\mathrm{i}2\pi k}{N}} \mathrm{e}^{\frac{\mathrm{i}2\pi kn'}{N/2}}\right) = \\
&= \mathrm{FHYT}(N/2,\mathbf{y})_{k \bmod N/2} + \sum_{n'=0}^{N/2-1} \mathbf{z}_{n'} \,\mathrm{ri}\left(\mathrm{e}^{\frac{\mathrm{i}2\pi k}{N}} \mathrm{e}^{\frac{\mathrm{i}2\pi kn'}{N/2}}\right) .
\end{aligned}$$

Zostáva už len vyjadriť poslednú sumu pomocou FHYT. Využijeme pritom rovnosť

$$2\mathrm{e}^{\mathrm{i}\alpha} = (1+\mathrm{i})\cos\alpha + (1-\mathrm{i})\cos -\alpha .$$

Takže dostávame:

$$\begin{aligned}
&\sum_{n'=0}^{N/2-1} \mathbf{z}_{n'} \operatorname{ri}\left(\mathrm{e}^{\frac{\mathrm{i}2\pi k}{N}} \mathrm{e}^{\frac{\mathrm{i}2\pi k n'}{N/2}}\right) = \\
&\quad = \frac{1}{2} \sum_{n'=0}^{N/2-1} \mathbf{z}_{n'} \operatorname{ri}\left[\left(\cos\tfrac{2\pi k}{N} + \mathrm{i}\sin\tfrac{2\pi k}{N}\right)\left((1+\mathrm{i})\operatorname{cas}\tfrac{2\pi k n'}{N/2} + (1-\mathrm{i})\operatorname{cas}\tfrac{-2\pi k n'}{N/2}\right)\right] = \\
&\quad = \sum_{n'=0}^{N/2-1} \mathbf{z}_{n'} \left[\cos\tfrac{2\pi k}{N}\operatorname{cas}\tfrac{2\pi k n'}{N/2} + \sin\tfrac{2\pi k}{N}\operatorname{cas}\tfrac{-2\pi k n'}{N/2}\right] = \\
&\quad = \widehat{\mathbf{z}}_k \cos\tfrac{2\pi k}{N} + \widehat{\mathbf{z}}_{-k} \sin\tfrac{2\pi k}{N} \,,
\end{aligned}$$

kde $\widehat{\mathbf{z}}_k = \mathrm{FHYT}(N/2, \mathbf{z})_{k \bmod N/2}$. Teda pre $k \in \{0, \ldots, N/2\}$:

$$\begin{aligned}
\mathrm{FHYT}(N, \mathbf{x})_k &= \widehat{\mathbf{y}}_k + \widehat{\mathbf{z}}_k \cos\tfrac{2\pi k}{N} + \widehat{\mathbf{z}}_{-k} \sin\tfrac{2\pi k}{N} \,, \\
\mathrm{FHYT}(N, \mathbf{x})_{k+N/2} &= \widehat{\mathbf{y}}_k - \widehat{\mathbf{z}}_k \cos\tfrac{2\pi k}{N} - \widehat{\mathbf{z}}_{-k} \sin\tfrac{2\pi k}{N} \,,
\end{aligned}$$

kde

$$\begin{aligned}
\widehat{\mathbf{y}}_k &= \mathrm{FHYT}(N/2, \mathbf{y})_{k \bmod N/2} \,, \\
\widehat{\mathbf{z}}_k &= \mathrm{FHYT}(N/2, \mathbf{z})_{k \bmod N/2} \,.
\end{aligned}$$

Ak N je celočíselná mocnina čísla 2, môžeme túto metódu aplikovať rekurzívne, až pokým sa nedostaneme ku triviálnej jednobodovej transformácii:

$$\mathrm{FHYT}(1, \mathbf{x})_k = \sum_{n=0}^{0} \mathbf{x}_n \operatorname{cas}\left(\tfrac{2\pi k n}{N}\right) = \mathbf{x}_0 \,.$$

1.2.2. Rekurzívna verzia algoritmu FHYT a jeho časová zložitosť

Na základe týchto výsledkov môžeme teraz implementovať algoritmus FHYT (v hypotetickom programovacom jazyku podobnom Pascalu, ktorý umožňuje použitie polí ako argumentov a výsledkov funkcií).

Časovú zložitosť algoritmu FHYT vyjadríme v celkovom počte vykonaných operácií (napr. aritmetických). Nech toto číslo je $T(p)$ pre transformáciu postupnosti pozostávajúcej z 2^{p-1} bodov. Ak sa bližšie pozrieme na procedúru FHYT (obr. 1.1), môžeme ľahko vidieť, že $T(p)$ musí spĺňať nasledujúci rekurentný vzťah:

$$\begin{aligned}
T(0) &= c \,, \\
T(p+1) &= 2T(p) + d2^p \,,
\end{aligned}$$

kde c a d sú nejaké konštanty[6]. Riešením tohto vzťahu je

$$T(p) = 2^{p-1}(2c + pd)$$

[6] Ak by sme napríklad pod operáciami chápali iba násobenie reálnych čísel, konštanta c by bola rovná 1 a konštanta d by mala hodnotu 2.

```
RADIX2-FHYT (N: integer, x: array of real)
1  variables
2     y,z,X,Y,Z: array of real
3     k,n',N',t: integer
4  if N = 1
5     then return x                                  ▷ triviálny prípad, keď N = 1
6     else begin                                     ▷ vykonanie dvoch menších FHYT
7        N' ← N/2                                    ▷ ich veľkosť
8        for n' ← 0 to N'−1 do begin                 ▷ rozdelenie na dve časti
9           y[n'] ← x[2·n']
10          z[n'] ← x[2·n'+1]
11       end
12       Y ← RADIX2-FHYT(N',y)                       ▷ párna časť
13       Z ← RADIX2-FHYT(N',z)                       ▷ nepárna časť
14       for k ← 0 to N'−1 do begin
15          t ← Z[k]·cos(2πk/N) + Z[−k mod N']·sin(2πk/N)
16          X[k] ← Y[k] + t
17          X[k+N'] ← Y[k] − t
18       end
19       return X
20    end
```

Obr. 1.1: Rýchly algoritmus výpočtu diskrétnej Hartleyho transformácie. Hodnota N vyjadruje počet prvkov postupnosti **x**.

alebo, vyjadrené pomocou dĺžky $N(=2^{p-1})$ vstupnej postupnosti:

$$T'(N) = 2N(2c + d(1+\lg N)) = \Theta(N \lg N) .$$

Teda časová zložitosť algoritmu FHYT je asymptoticky rovná časovej zložitosti algoritmu FFT.

1.3. Diskrétna cas-cas transformácia

Diskrétna cas-cas transformácia je dvojrozmernou analógiou diskrétnej Hartleyho transformácie. Je to transformácia so separovateľným jadrom, a teda je možné ju vypočítať použitím diskrétnej Hartleyho transformácie. Podobne ako jednorozmerná Hartleyho transformácia je sama k sebe inverzná, aj cas-cas transformácia má túto vlastnosť.

Definícia 1.7 (Diskrétna cas-cas transformácia).
Nech **x** je matica rozmerov $N_1 \times N_2$. *Diskrétna cas-cas transformácia* (DCCT) matice **x** je matica $\bar{h}_{2D}(\mathbf{x})$ rozmerov $N_1 \times N_2$, ktorej prvky sú definované:

$$\bar{h}_{2D}(\mathbf{x})_{k_1,k_2} = \frac{1}{\sqrt{N_1 N_2}} \sum_{k_1=0}^{N_1-1} \sum_{k_2=0}^{N_2-1} \mathbf{x}_{k_1,k_2} \operatorname{cas}\left(\frac{2\pi k_1 n_1}{N_1}\right) \operatorname{cas}\left(\frac{2\pi k_2 n_2}{N_2}\right) .$$

Inverzná diskrétna cas-cas transformácia je daná tým istým vzťahom:

$$\mathbf{x}_{n_1,n_2} = \bar{h}_{2\mathrm{D}}^{-1}(\bar{h}_{2\mathrm{D}}(\mathbf{x}))_{n_1,n_2} = \frac{1}{\sqrt{N_1 N_2}} \sum_{k_1=0}^{N_1-1} \sum_{k_2=0}^{N_2-1} \bar{h}_{2\mathrm{D}}(\mathbf{x})_{k_1,k_2} \operatorname{cas}\left(\tfrac{2\pi k_1 n_1}{N_1}\right) \operatorname{cas}\left(\tfrac{2\pi k_2 n_2}{N_2}\right) .$$

1.3.1. Výpočet diskrétnej cas-cas transformácie

Malo by byť na prvý pohľad zrejmé, že dvojrozmernú diskrétnu cas-cas transformáciu možno vypočítať použitím nezávislých diskrétnych Hartleyho transformácií[7] na riadky matice a následne na stĺpce novovzniknutej matice, resp. v opačnom poradí, t.j. najprv na stĺpce a potom na riadky.

Aby sme sa o tom presvedčili, uvažujme takéto preusporiadanie definície DCCT:

$$\bar{h}_{2\mathrm{D}}(\mathbf{x})_{k_1,k_2} = \mathbf{x}''_{k_1,k_2} = \frac{1}{\sqrt{N_1}} \sum_{n_1=0}^{N_1-1} \left[\frac{1}{\sqrt{N_2}} \sum_{n_2=0}^{N_2-1} \mathbf{x}_{n_1,n_2} \operatorname{cas}\left(\tfrac{2\pi k_2 n_2}{N_2}\right) \right] \operatorname{cas}\left(\tfrac{2\pi k_1 n_1}{N_1}\right) ,$$

alebo prehľadnejšie:

$$\mathbf{x}''_{k_1,k_2} = \frac{1}{\sqrt{N_1}} \sum_{n_1=0}^{N_1-1} \mathbf{x}'_{n_1,k_2} \operatorname{cas}\left(\tfrac{2\pi k_1 n_1}{N_1}\right) , \quad \text{kde} \quad \mathbf{x}'_{n_1,k_2} = \frac{1}{\sqrt{N_2}} \sum_{n_2=0}^{N_2-1} \mathbf{x}_{n_1,n_2} \operatorname{cas}\left(\tfrac{2\pi k_2 n_2}{N_2}\right) .$$

Oba tieto výrazy sú diskrétne Hartleyho transformácie. Samozrejme, nezáleží na tom, či vyhodnotíme DHYT najskôr pre riadky a potom pre stĺpce alebo naopak. Tento postup je zrejme možné zovšeobecniť pre ľubovoľnú diskrétnu transformáciu so separovateľným jadrom akejkoľvek dimenzie.

Časová zložitosť

Ak by sme predpokladali, že N_1 a N_2 sú celočíselné mocniny 2, mohli by sme použiť Radix 2 FHYT algoritmus pre výpočet jednorozmerných DHYT. Počet $T'(N)$ aritmetických operácií, ktoré vykoná algoritmus na výpočet N-bodovej DHYT, bol odvodený vyššie:

$$T'(N) = \Theta(N \lg N) ,$$

takže počet týchto operácií, ktoré vykoná algoritmus pre výpočet DCCT používajúci DHYT na riadky a stĺpce je daný:

$$\begin{aligned} T''(N_1,N_2) &= N_1 T'(N_2) + N_2 T'(N_1) = N_1 \Theta(N_2 \lg N_2) + N_2 \Theta(N_1 \lg N_1) = \\ &= \Theta(N_1 N_2 \lg(N_1 N_2)) . \end{aligned}$$

Teda takýto algoritmus pre maticu veľkosti $N_1 \times N_2$ vykoná asymptoticky rovnaký počet aritmetických operácií ako FHYT algoritmus pre vektor dĺžky $N_1 N_2$.

[7] V prípade dvojrozmernej DFT to zrejme budú DFT.

Kapitola 2

Diskrétna konvolúcia

Cieľom tejto kapitoly je ukázať existenciu rýchleho algoritmu pre výpočet diskrétnej konvolúcie dvojrozmerných postupností. Najprv odvodíme algoritmus využívajúci vlastnosti diskrétnej Fourierovej transformácie, neskôr ukážeme možnosť jeho ďalšej optimalizácie použitím diskrétnej cas-cas transformácie, ktorá narozdiel od Fourierovej transformácie môže pracovať čisto s reálnymi číslami.

V nasledujúcom sa budeme zaoberať výpočtom konvolúcie reálnych postupností, ktoré sú nenulové na ohraničenom intervale. Ukážeme veľmi efektívny spôsob výpočtu konvolúcie takýchto postupností použitím rýchlej Fourierovej transformácie a rýchlej Hartleyho transformácie.

2.1. Jednorozmerná diskrétna konvolúcia

Definícia 2.1 (Konvolúcia postupností).
Nech $\mathbf{x}$ a $\mathbf{y}$ sú dve komplexné postupnosti (funkcie zo $\mathbb{Z}$ do $\mathbb{C}$). *Konvolúcia* týchto postupností je postupnosť $\mathbf{x} * \mathbf{y}$, ktorej členy sú definované nasledujúcim vzťahom:

$$(\mathbf{x} * \mathbf{y})_n = \sum_{k \in \mathbb{Z}} \mathbf{x}_k \mathbf{y}_{n-k} \, .$$

Poznámka.
Nie je ťažké ukázať, že konvolúcia komutuje, t.j. že $\mathbf{x} * \mathbf{y} = \mathbf{y} * \mathbf{x}$.

Lema 2.2 (O invariantnosti diskrétnej konvolúcie voči posunutiu).
Nech $\mathbf{x}$ a $\mathbf{y}$ sú dve postupnosti. Nech x_s a y_s sú prirodzené čísla. Nech $\mathbf{x}'$ a $\mathbf{y}'$ sú také postupnosti, pre ktoré $\mathbf{x}'_n = \mathbf{x}_{n+x_s}$ a $\mathbf{y}'_n = \mathbf{y}_{n+y_s}$. Potom $(\mathbf{x} * \mathbf{y})_n = (\mathbf{x}' * \mathbf{y}')_{n-(x_s+y_s)}$.

Dôkaz. Priamočiaro vyjadrením $(\mathbf{x}' * \mathbf{y}')_{n-(x_s+y_s)}$:

$$\begin{aligned}
(\mathbf{x}' * \mathbf{y}')_{n-(x_s+y_s)} &= \sum_{k \in \mathbb{Z}} \mathbf{x}'_k \mathbf{y}'_{n-(x_s+y_s)-k} = \sum_{k \in \mathbb{Z}} \mathbf{x}_{k+x_s} \mathbf{y}_{n-(x_s+y_s)-k+y_s} = \\
&= \sum_{k \in \mathbb{Z}} \mathbf{x}_{k+x_s} \mathbf{y}_{n-(k+x_s)} = \sum_{k' \in \mathbb{Z}} \mathbf{x}_{k'} \mathbf{y}_{n-k'} = \\
&= (\mathbf{x} * \mathbf{y})_n \, .
\end{aligned}$$

■

Definícia 2.3 (Cyklická konvolúcia vektorov).
Pre dvojicu vektorov rovnakej dĺžky (N) definujeme operáciu *cyklickej konvolúcie* ako vektor $\mathbf{x} \otimes \mathbf{y}$ dĺžky N, ktorého členy nadobúdajú nasledujúce hodnoty:

$$(\mathbf{x} \otimes \mathbf{y})_n = \sum_{k=0}^{N-1} \mathbf{x}_k \mathbf{y}_{(n-k) \bmod N} \,.$$

Nasledujúca lema a veta hovoria o možnosti výpočtu diskrétnej konvolúcie dvoch konečných postupností pomocou cyklickej konvolúcie.

Lema 2.4.
Nech $\mathbf{x}$ a $\mathbf{y}$ sú postupnosti, ktoré sú nulové mimo intervalov $\{0,\ldots,x_e\}$, kde $x_e \geq 0$, resp. $\{0,\ldots,y_e\}$, kde $y_e \geq 0$. Nech $N = x_e + y_e + 1$ a nech $\mathbf{x}'$ a $\mathbf{y}'$ sú vektory dĺžky N, definované $\mathbf{x}'_n = \mathbf{x}_n$ a $\mathbf{y}'_n = \mathbf{y}_n$ pre $n \in \{0,\ldots,N-1\}$. Potom, za predpokladu, že cyklickú konvolúciu chápeme ako operáciu na vektoroch dĺžky N, platí

$$(\mathbf{x} * \mathbf{y})_n = \begin{cases} (\mathbf{x}' \otimes \mathbf{y}')_n \,, & \text{ak } n \in \{0,\ldots,x_e+y_e\} \,, \\ 0 \,, & \text{inak.} \end{cases}$$

Dôkaz. Rozoberieme postupne všetky tri prípady, ktoré môžu nastať: $n < 0$, $n \in \{0,\ldots,x_e+y_e\}$ a $n > x_e + y_e$.

1. Nech $n < 0$. Zrejme pre každé $k \in \mathbb{Z}$ platí $\mathbf{x}_k \mathbf{y}_{n-k} = 0$. Ak totiž $k < 0$, je $\mathbf{x}_k = 0$. Ak naopak $k \geq 0$, platí $n - k < 0$, z čoho zase $\mathbf{y}_{n-k} = 0$. Teda

   $$(\mathbf{x} * \mathbf{y})_n = \sum_{k \in \mathbb{Z}} \mathbf{x}_k \mathbf{y}_{n-k} = \sum_{k \in \mathbb{Z}} 0 = 0 \,.$$

2. Nech $n \in \{0,\ldots,x_e+y_e\}$. Počítajme, čomu sa rovná výraz $(\mathbf{x}' \otimes \mathbf{y}')_n$:

   $$(\mathbf{x}' \otimes \mathbf{y}')_n = \sum_{k=0}^{N-1} \mathbf{x}'_k \mathbf{y}'_{(n-k) \bmod N} \,.$$

   Keďže $\mathbf{x}'_k = 0$ pre $k > x_e$, zredukujeme oblasť sumácie:

   $$(\mathbf{x}' \otimes \mathbf{y}')_n = \sum_{k=0}^{x_e} \mathbf{x}'_k \mathbf{y}'_{(n-k) \bmod N} \,.$$

   Teraz môžu nastať dva prípady:

   - Nech $k \leq n$. Potom $n - k \in \{0,\ldots,x_e+y_e\}$, čiže $(n-k) \bmod N = n-k$, čo v konečnom dôsledku znamená

     $$\mathbf{y}'_{(n-k) \bmod N} = \mathbf{y}_{n-k} \,.$$

- Nech teraz $k > n$. Zrejme $-x_e < n-k < 0$, z čoho vyplýva $(n-k) \bmod N \in \{y_e + 1, \ldots, x_e + y_e\}$. Nakoľko pre $m > y_e$ je $\mathbf{y}'_m$ rovné 0, dostávame $\mathbf{y}'_{(n-k) \bmod N} = 0$. Z predpokladov vety vyplýva, že $\mathbf{y}_{n-k} = 0$ pre $n-k < 0$. Výsledná rovnosť teda je

$$y'_{(n-k) \bmod N} = 0 = \mathbf{y}_{n-k} \,.$$

3. Nech $n > x_e + y_e$. Aby y_{n-k} bolo nenulové, musí byť $n-k \in \{0, \ldots, y_e\}$, teda k musí byť väčšie ako x_e. Pre takéto k je ale $x_k = 0$. Zrejme pre každé $k \in \mathbb{Z}$ platí $\mathbf{x}_k \mathbf{y}_{n-k} = 0$. Teda

$$(\mathbf{x} * \mathbf{y})_n = \sum_{k \in \mathbb{Z}} \mathbf{x}_k \mathbf{y}_{n-k} = \sum_{k \in \mathbb{Z}} 0 = 0 \,.$$

■

Veta 2.5 (Výpočet diskrétnej konvolúcie pomocou cyklickej konvolúcie).
Nech $\mathbf{x}$ a $\mathbf{y}$ sú postupnosti, ktoré sú nulové mimo intervalov $\{x_s, \ldots, x_e\}$, kde $x_e \geq x_s$, resp. $\{y_s, \ldots, y_e\}$, kde $y_e \geq y_s$. Nech $N = x_e + y_e - (x_s + y_s) + 1$ a nech $\mathbf{x}'$ a $\mathbf{y}'$ sú vektory dĺžky N, definované $\mathbf{x}'_n = \mathbf{x}_{n+x_s}$ a $\mathbf{y}'_n = \mathbf{y}_{n+y_s}$ pre $n \in \{0, \ldots, N-1\}$. Potom, za predpokladu, že cyklickú konvolúciu chápeme ako operáciu na vektoroch dĺžky N, platí

$$(\mathbf{x} * \mathbf{y})_n = \begin{cases} (\mathbf{x}' \otimes \mathbf{y}')_{n-(x_s+y_s)} \,, & \text{ak } n \in \{x_s + y_s, \ldots, x_e + y_e\} \,, \\ 0 \,, & \text{inak.} \end{cases}$$

Dôkaz. Vyplýva priamo z liem 2.2 a 2.4. ■

Uvedená veta vyjadruje hodnoty diskrétnej konvolúcie dvoch v podstate nulových (nulových s výnimkou konečného počtu bodov) postupností na základe hodnôt cyklickej konvolúcie podpostupností týchto postupností.

Nasledujúca veta hovorí o možnosti výpočtu cyklickej konvolúcie pomocou diskrétnej Fourierovej transformácie. Jej priamym dôsledkom je existencia algoritmu na výpočet cyklickej konvolúcie, ktorého časová zložitosť je asymptoticky rovná časovej zložitosti algoritmu FFT.

Veta 2.6 (Výpočet cyklickej konvolúcie pomocou DFT).
Nech $\mathbf{x}$ a $\mathbf{y}$ sú komplexné vektory dĺžky $N > 1$. Potom platí

$$\mathbf{x} \otimes \mathbf{y} = \sqrt{N}\, \Phi^{-1}(\Phi(\mathbf{x}) \cdot \Phi(\mathbf{y})) \,,$$

kde $\Phi(\mathbf{x}) \cdot \Phi(\mathbf{y})$ je súčin po zložkách.

Dôkaz. Vezmeme $\Phi(\mathbf{x} \otimes \mathbf{y})$, rozvinieme podľa definície diskrétnej Fourierovej transformácie a následne $\mathrm{e}^{-\mathrm{i}2\pi mn/N}$ rozložíme na súčin $\mathrm{e}^{-\mathrm{i}2\pi mk/N} \mathrm{e}^{-\mathrm{i}2\pi m(n-k)/N}$:

$$\Phi(\mathbf{x} \otimes \mathbf{y})_m = \frac{1}{\sqrt{N}} \sum_{n=0}^{N-1} (\mathbf{x} \otimes \mathbf{y})_n \mathrm{e}^{\frac{-\mathrm{i}2\pi mn}{N}} = \frac{1}{\sqrt{N}} \sum_{n=0}^{N-1} \left[\sum_{k=0}^{N-1} \mathbf{x}_k \mathbf{y}_{(n-k) \bmod N} \right] \mathrm{e}^{\frac{-\mathrm{i}2\pi mn}{N}} =$$

$$= \frac{1}{\sqrt{N}} \sum_{n=0}^{N-1} \left[\sum_{k=0}^{N-1} \mathbf{x}_k \mathbf{y}_{(n-k) \bmod N} \right] \mathrm{e}^{\frac{-\mathrm{i}2\pi mk}{N}} \mathrm{e}^{\frac{-\mathrm{i}2\pi m(n-k)}{N}} \,.$$

Posledný výraz mierne preusporiadame zámenou poradia sumácie:

$$\Phi(\mathbf{x}\otimes\mathbf{y})_m = \frac{1}{\sqrt{N}}\sum_{k=0}^{N-1}\mathbf{x}_k \mathrm{e}^{\frac{-\mathrm{i}2\pi mk}{N}}\left[\sum_{n=0}^{N-1}\mathbf{y}_{(n-k)\bmod N}\mathrm{e}^{\frac{-\mathrm{i}2\pi m(n-k)}{N}}\right] .$$

Ďalší „trik“ spočíva vo využití faktu, že vnútorná sumácia je nezávislá na k vďaka skutočnosti, že $\mathbf{y}_{(n-k)\bmod N}$ aj $\mathrm{e}^{-\mathrm{i}2\pi m(n-k)/N}$ sú periodické funkcie parametra n s periódou práve N. Teraz môžeme uskutočniť náhradu výrazu $n-k$ za n a vnútornú sumu vyňať mimo vonkajšej.

$$\begin{aligned}\Phi(\mathbf{x}\otimes\mathbf{y})_m &= \frac{1}{\sqrt{N}}\left[\sum_{k=0}^{N-1}\mathbf{x}_k\mathrm{e}^{\frac{-\mathrm{i}2\pi mk}{N}}\right]\left[\sum_{n=0}^{N-1}\mathbf{y}_n\mathrm{e}^{\frac{-\mathrm{i}2\pi mn}{N}}\right] = \\ &= \sqrt{N}\left[\frac{1}{\sqrt{N}}\sum_{k=0}^{N-1}\mathbf{x}_k\mathrm{e}^{\frac{-\mathrm{i}2\pi mk}{N}}\right]\left[\frac{1}{\sqrt{N}}\sum_{n=0}^{N-1}\mathbf{y}_n\mathrm{e}^{\frac{-\mathrm{i}2\pi mn}{N}}\right] .\end{aligned}$$

Po aplikovaní operátora inverznej DFT (s využitím jeho lineárnosti) na obe strany rovnosti dostávame

$$\mathbf{x}\otimes\mathbf{y} = \sqrt{N}\,\Phi^{-1}(\Phi(\mathbf{x})\cdot\Phi(\mathbf{y})) ,$$

čo je práve dokazované tvrdenie. ■

2.1.1. Časová zložitosť výpočtu jednorozmernej diskrétnej konvolúcie

Nech $\mathbf{x}$ a $\mathbf{y}$ sú reálne postupnosti, ktoré sú nulové mimo intervalov $\{x_s,\ldots,x_e\}$, kde $x_e \geq$ $\geq x_s$, resp. $\{y_s,\ldots,y_e\}$, kde $y_e \geq y_s$. Nech $N = x_e + y_e - (x_s + y_s) + 1$. Na základe vety 2.5 môžeme v čase $O(N)$ tento problém transformovať na problém výpočtu cyklickej konvolúcie postupností dĺžky N. Z vety 2.6 a lemy 1.2 vyplýva, že cyklickú konvolúciu postupnosti dĺžky N možno vypočítať pomocou diskrétnej Fourierovej transformácie, pre ktorú sú známe algoritmy (viď napr. [7], [13], [16]) s časovou zložitosťou $O(N\lg N)$. Druhou možnosťou je namiesto DFT použiť diskrétnu Hartleyho transformáciu (treba však využiť lemu 1.6), ktorá má rovnakú asymptotickú zložitosť, avšak pracuje iba s reálnymi číslami, čím je možné zredukovať počet potrebných operácií približne na polovicu. Teda diskrétnu konvolúciu postupností $\mathbf{x}$ a $\mathbf{y}$ možno vypočítať v čase $O(N\lg N)$.

2.2. Dvojrozmerná diskrétna konvolúcia

Definícia 2.7 (Dvojrozmerná konvolúcia).
Nech $\mathbf{x}$ a $\mathbf{y}$ sú dvojrozmerné komplexné postupnosti (funkcie zo $\mathbb{Z}^2$ do $\mathbb{C}$). *Konvolúcia* týchto postupností je postupnosť $\mathbf{x}*\mathbf{y}$, ktorej členy sú definované nasledujúcim vzťahom:

$$(\mathbf{x}*\mathbf{y})_{n_1,n_2} = \sum_{k_1\in\mathbb{Z}}\sum_{k_2\in\mathbb{Z}}\mathbf{x}_{k_1,k_2}\mathbf{y}_{n_1-k_1,n_2-k_2} .$$

Definícia 2.8 (Dvojrozmerná cyklická konvolúcia).
Nech $\mathbf{x}$ a $\mathbf{y}$ sú dve matice rozmerov $N_1 \times N_2$. Ich cyklickú konvolúciu definujeme ako maticu $\mathbf{x} \otimes \mathbf{y}$ rozmerov $N_1 \times N_2$, ktorej prvky sú dané:

$$(\mathbf{x} \otimes \mathbf{y})_{n_1,n_2} = \sum_{k_1=0}^{N_1-1} \sum_{k_2=0}^{N_2-1} \mathbf{x}_{k_1,k_2} \mathbf{y}_{(n_1-k_1) \bmod N_1, (n_2-k_2) \bmod N_2} \,.$$

Veta 2.9 (Výpočet 2D diskrétnej konvolúcie pomocou 2D cyklickej konvolúcie).
Nech $\mathbf{x}$ a $\mathbf{y}$ sú dvojrozmerné postupnosti, ktoré sú nulové mimo obdĺžnikov $\{x_s^{(1)}, \ldots, x_e^{(1)}\} \times \times \{x_s^{(2)}, \ldots, x_e^{(2)}\}$, kde $x_e^{(1)} \geq x_s^{(1)}$ a $x_e^{(2)} \geq x_s^{(2)}$, resp. $\{y_s^{(1)}, \ldots, y_e^{(1)}\} \times \{y_s^{(2)}, \ldots, y_e^{(2)}\}$, kde $y_e^{(1)} \geq \geq y_s^{(1)}$ a $y_e^{(2)} \geq y_s^{(2)}$. Nech $N_1 = x_e^{(1)} + y_e^{(1)} - (x_s^{(1)} + y_s^{(1)}) + 1$ a $N_2 = x_e^{(2)} + y_e^{(2)} - (x_s^{(2)} + y_s^{(2)}) + 1$ a nech $\mathbf{x}'$ a $\mathbf{y}'$ sú matice veľkosti $N_1 \times N_2$, definované

$$\mathbf{x}'_{n_1,n_2} = \mathbf{x}_{n_1+x_s^{(1)}, n_2+x_s^{(2)}} \quad \text{a} \quad \mathbf{y}'_{n_1,n_2} = \mathbf{y}_{n_1+y_s^{(1)}, n_2+y_s^{(2)}}$$

pre $n_1 \in \{0, \ldots, N_1 - 1\}$ a $n_2 \in \{0, \ldots, N_2 - 1\}$. Potom, za predpokladu, že cyklickú konvolúciu chápeme ako operáciu na maticiach veľkosti $N_1 \times N_2$, platí

$$(\mathbf{x} * \mathbf{y})_{n_1,n_2} = \begin{cases} (\mathbf{x}' \otimes \mathbf{y}')_{n_1-(x_s^{(1)}+y_s^{(1)}), n_2-(x_s^{(2)}+y_s^{(2)})} \,, & \text{ak } n_i \in \{x_s^{(i)} + y_s^{(i)}, \ldots, x_e^{(i)} + y_e^{(i)}\} \,, \\ 0 \,, & \text{inak,} \end{cases}$$

kde $i \in \{1, 2\}$.

Dôkaz. Dôkaz tejto vety môže byť vedený rovnakým spôsobom ako v prípade jednorozmernej konvolúcie (viď dôkaz vety 2.5), a preto nemá hlbší zmysel sa ním bližšie zaoberať. ■

Veta 2.10 (Výpočet dvojrozmernej cyklickej konvolúcie pomocou 2D-DFT).
Nech $\mathbf{x}$ a $\mathbf{y}$ sú komplexné matice rozmeru $N_1 \times N_2$, pričom $N_1 > 1, N_2 > 1$. Potom platí

$$\mathbf{x} \otimes \mathbf{y} = \sqrt{N_1 N_2}\, \Phi_{2\mathrm{D}}^{-1}(\Phi_{2\mathrm{D}}(\mathbf{x}) \cdot \Phi_{2\mathrm{D}}(\mathbf{y})) \,,$$

kde $\Phi_{2\mathrm{D}}(\mathbf{x}) \cdot \Phi_{2\mathrm{D}}(\mathbf{y})$ je súčin po zložkách.

Dôkaz. Aby sme zjednodušili zápisy zavedieme označenie $\{a\}$ pre $\mathrm{e}^{-\mathrm{i}2\pi a}$ a pri indexácii prvkov matice budeme uvažovať jej nekonečné periodické rozšírenie, t.j. napríklad výraz $\mathbf{y}_{(n_1-k_1) \bmod N_1, (n_2-k_2) \bmod N_2}$ budeme skrátene zapisovať ako $\mathbf{y}_{n_1-k_1, n_2-k_2}$.

Prvým krokom je rozvinutie výrazu $\sqrt{N_1 N_2}\, \Phi_{2\mathrm{D}}(\mathbf{x} \otimes \mathbf{y})$ podľa definície dvojrozmernej diskrétnej Fourierovej transformácie. Ďalej dosadíme definíciu dvojrozmernej cyklickej konvolúcie a výraz $\left\{\frac{m_j n_j}{N_j}\right\}$ rozložíme na súčin $\left\{\frac{m_j k_j}{N_j}\right\} \left\{\frac{m_j (n_j - k_j)}{N_j}\right\}$. Podobne ako v jednorozmernom prípade, aj teraz uskutočníme zámenu poradia sumácie a následne využijeme periodicitu funkcií

$\mathbf{y}_{(n_1-k_1) \bmod N_1,(n_2-k_2) \bmod N_2}$ a $\left\{\frac{m_1(n_1-k_1)}{N_1}\right\}\left\{\frac{m_2(n_2-k_2)}{N_2}\right\}$ vzhľadom na oba argumenty n_1, n_2.

$$\begin{aligned}
&\sqrt{N_1N_2}\,\Phi_{2\mathrm{D}}(\mathbf{x}\otimes\mathbf{y})_{m_1,m_2} = \\
&\quad= \sum_{n_1=0}^{N_1-1}\sum_{n_2=0}^{N_2-1}(\mathbf{x}\otimes\mathbf{y})_{n_1,n_2}\left\{\frac{m_1n_1}{N_1}\right\}\left\{\frac{m_2n_2}{N_2}\right\} = \\
&\quad= \sum_{n_1=0}^{N_1-1}\sum_{n_2=0}^{N_2-1}\left[\sum_{k_1=0}^{N_1-1}\sum_{k_2=0}^{N_2-1}\mathbf{x}_{k_1,k_2}\mathbf{y}_{n_1-k_1,n_2-k_2}\right]\left\{\frac{m_1n_1}{N_1}\right\}\left\{\frac{m_2n_2}{N_2}\right\} = \\
&\quad= \sum_{n_1=0}^{N_1-1}\sum_{n_2=0}^{N_2-1}\left[\sum_{k_1=0}^{N_1-1}\sum_{k_2=0}^{N_2-1}\mathbf{x}_{k_1,k_2}\mathbf{y}_{n_1-k_1,n_2-k_2}\right]\left\{\frac{m_1k_1}{N_1}\right\}\left\{\frac{m_1(n_1-k_1)}{N_1}\right\}\left\{\frac{m_2k_2}{N_2}\right\}\left\{\frac{m_2(n_2-k_2)}{N_2}\right\} = \\
&\quad= \sum_{k_1=0}^{N_1-1}\sum_{k_2=0}^{N_2-1}\mathbf{x}_{k_1,k_2}\left\{\frac{m_1k_1}{N_1}\right\}\left\{\frac{m_2k_2}{N_2}\right\}\left[\sum_{n_1=0}^{N_1-1}\sum_{n_2=0}^{N_2-1}\mathbf{y}_{n_1-k_1,n_2-k_2}\left\{\frac{m_1(n_1-k_1)}{N_1}\right\}\left\{\frac{m_2(n_2-k_2)}{N_2}\right\}\right] = \\
&\quad= \left[\sum_{k_1=0}^{N_1-1}\sum_{k_2=0}^{N_2-1}\mathbf{x}_{k_1,k_2}\left\{\frac{m_1k_1}{N_1}\right\}\left\{\frac{m_2k_2}{N_2}\right\}\right]\left[\sum_{n_1=0}^{N_1-1}\sum_{n_2=0}^{N_2-1}\mathbf{y}_{n_1,n_2}\left\{\frac{m_1n_1}{N_1}\right\}\left\{\frac{m_2n_2}{N_2}\right\}\right] .
\end{aligned}$$

Posledným krokom dôkazu je upravenie oboch súm posledného výrazu na tvar z definície dvojrozmernej DFT a následná aplikácia inverznej dvojrozmernej DFT na obe strany rovnosti. Teda dostávame

$$\mathbf{x}\otimes\mathbf{y} = \sqrt{N_1N_2}\,\Phi_{2\mathrm{D}}^{-1}(\Phi_{2\mathrm{D}}(\mathbf{x})\cdot\Phi_{2\mathrm{D}}(\mathbf{y})) ,$$

čo je presne rovnosť z tvrdenia vety. ■

Poznámka.
Príklad rýchleho algoritmu počítajúceho dvojrozmernú cyklickú konvolúciu matíc **x**, **y** veľkosti $N_1 \times N_2$ je uvedený na obrázku 2.1.

Veta 2.11 (O rýchlom spôsobe výpočtu dvojrozmernej diskrétnej konvolúcie).
Nech **x** a **y** sú dvojrozmerné postupnosti nulové mimo intervalov z vety 2.9. Potom ich diskrétnu konvolúciu možno vypočítať v čase $O(N_1N_2 \lg N_1N_2)$, kde N_1 a N_2 sú konštanty z vety 2.9.

Dôkaz (neformálny). Výsledkom operácie konvolúcie dvojrozmerných postupností **x** a **y** je podľa vety 2.9 dvojrozmerná postupnosť nulová mimo obdĺžnika veľkosti $N_1 \times N_2$, pričom hodnoty v tomto obdĺžniku možno vypočítať priamo z postupností **x** a **y** pomocou dvojrozmernej cyklickej konvolúcie, ktorú na základe vety 2.10 a lemy 1.4 možno vypočítať trojnásobným použitím algoritmu dvojrozmernej diskrétnej Fourierovej transformácie matíc veľkosti $N_1 \times N_2$. Existujú algoritmy, pomocou ktorých je možné vypočítať dvojrozmernú DFT matíc tejto veľkosti v čase $\Theta(N_1N_2 \lg N_1N_2)$ (viď napr. [21], [33], [34]). Počet všetkých zvyšných operácií, ktoré musí algoritmus na výpočet dvojrozmernej diskrétnej konvolúcie, je rovný $\Theta(N_1N_2)$, čo je zrejme zanedbateľné (z asymptotického hľadiska) v porovnaní s $\Theta(N_1N_2 \lg N_1N_2)$. Z uvedeného priamo vyplýva, že dvojrozmernú diskrétnu konvolúciu možno vypočítať v čase $O(N_1N_2 \lg N_1N_2)$. ■

FAST-CIRCULAR-2D-CONVOLUTION (N_1, N_2: integer, $\mathbf{x}, \mathbf{y}$: matrix of complex)

```
1  variables
2     k1, k2: integer
3     X, Y, Z, z: matrix of complex
4  X ← FFT2D(N1, N2, x)
5  Y ← FFT2D(N1, N2, y)
6  for k1 ← 0 to N1 − 1 do
7     for k2 ← 0 to N2 − 1 do
8        Z[k1, k2] ← X[k1, k2] · Y[k1, k2]
9  z ← FFT2D^{-1}(N1, N2, Z)
10 return z · √(N1 N2)
```

Obr. 2.1: Rýchly algoritmus výpočtu dvojrozmernej diskrétnej cyklickej konvolúcie. Hodnoty N_1, resp. N_2 vyjadrujú počet riadkov, resp. stĺpcov matíc **x** a **y**. Na základe lemy 1.4 možno na riadku č. 9 nahradiť s patričnými úpravami volanie $\mathrm{FFT}_{2\mathrm{D}}^{-1}$ volaním $\mathrm{FFT}_{2\mathrm{D}}$.

FAST-2D-CONVOLUTION (**x**, **y**: complex 2D sequence)

```
1  variables
2     mx, my, mz: matrix of complex
3     z: complex 2D sequence
4     width, height: integer
5  Z postupností x, y vyber najmenšie matice mx, my obsahujúce všetky nenulové prvky
6  width ← mx.width + my.width − 1
7  height ← mx.height + my.height − 1
8  Vytvor maticu mz veľkosti aspoň height × width.
9  Matice mx, my rozšír (doplnením nulami smerom doprava nadol) na veľkosť matice mz.
10 mz ← FAST-CIRCULAR-2D-CONVOLUTION(mz.height, mz.width, mx, my)
11 Z hodnôt matice mz vytvor 2D postupnosť z (viď vetu 2.9)
12 return z
```

Obr. 2.2: Rýchly algoritmus výpočtu dvojrozmernej diskrétnej konvolúcie. Tento algoritmus je vhodný iba na výpočet konvolúcie dvojrozmerných postupností, ktoré sú v postate nulové (t.j. s výnimkou konečného počtu bodov sú nulové). Jednoduchým spôsobom je možné ho rozšíriť aj na postupnosti, ktoré sú v podstate konštantné. Na riadku 5 môže vzniknúť menší problém, konkrétne keď je aspoň jedna z postupností **x**, **y** konštantne nulová. Vtedy stačí vziať z danej postupnosti ľubovoľný „výrez“.

Dôsledok 2.12.
Nech **x** a **y** sú dvojrozmerné postupnosti nulové mimo intervalov veľkosti $O(n) \times O(n)$. Potom ich diskrétnu konvolúciu možno vypočítať v čase $O(n^2 \lg n)$.

Poznámka.
Cyklickú konvolúciu dvoch matíc rozmeru $M \times N$ možno vypočítať pomocou diskrétnej Fourierovej transformácie použitím $O(MN \lg MN)$ komplexných operácií. Keď sú však obe matice reálne, výsledná konvolúcia je tiež reálna, takže neexistuje žiadny racionálny dôvod používať

v priebehu výpočtu imaginárne čísla. Namiesto DFT je možné použiť *dvojrozmernú diskrétnu cas-cas transformáciu*, pri ktorej sa trochu zmení násobenie vo frekvenčnej oblasti[1].

2.2.1. Výpočet dvojrozmernej cyklickej konvolúcie použitím cas-cas transformácie

Pri výpočte cyklickej konvolúcie $\mathbf{x} \otimes \mathbf{y}$ pomocou dvojrozmernej DFT sa predpokladá použitie dyadického násobenia matíc (po jednotlivých prvkoch). Výpočet konvolúcie pomocou DCCT je trochu komplikovanejší, ale dá sa vyjadriť na základe vzťahu medzi DCCT a 2D-DFT.

Predpokladajme, že existuje bijektívne zobrazenie F, pre ktoré platí:

$$\Phi_{2\mathrm{D}}(\mathbf{x}) = F(\overline{h}_{2\mathrm{D}}(\mathbf{x})) \quad \text{a} \quad \overline{h}_{2\mathrm{D}}(\mathbf{x}) = F^{-1}(\Phi_{2\mathrm{D}}(\mathbf{x}))\,, \tag{2.1}$$

pre každú reálnu maticu $\mathbf{x}$. Existenciu takéhoto zobrazenia ukážeme v dodatku A.

Zo vzťahu 2.1 vyplýva, že dvojrozmernú cyklickú konvolúciu možno počítať nasledujúcim spôsobom, ktorý je trochu komplikovanejší ako pri použití dvojrozmernej DFT, ale ako už bolo povedané, na výpočet diskrétnej cyklickej konvolúcie použitím diskrétnej cas-cas transformácie potrebujeme vykonať približne o polovicu menej aritmetických operácií (s reálnymi číslami):

$$\mathbf{x} \otimes \mathbf{y} = \sqrt{N_1 N_2}\,\overline{h}_{2\mathrm{D}}^{-1}(F^{-1}(F(\overline{h}_{2\mathrm{D}}(\mathbf{x})) \cdot F(\overline{h}_{2\mathrm{D}}(\mathbf{y}))))\,.$$

Hoci by mal byť postup zrejmý, pokúsme sa ešte o optimalizáciu algoritmu, mimochodom i preto, aby sme sa vyhli použitiu komplexnej aritmetiky. Pre skrátenie zápisu zaveďme ešte tzv. *operátor zrkadlenia*.

Definícia 2.13 (Operátor zrkadlenia).
Ak $\mathbf{x}$ je matica rozmeru $N_1 \times N_2$ a $a, b \in \{0,1\}$, potom nech výraz $\mu_{a,b}(\mathbf{x})$ označuje maticu $\mathbf{y}$, pre ktorú platí:

$$\mathbf{y}_{k_1,k_2} = \mathbf{x}_{((-1)^a k_1) \bmod N_1, ((-1)^b k_2) \bmod N_2}\,.$$

Počítajme teraz, čomu sa rovná $F^{-1}(F(\mathbf{x}) \cdot F(\mathbf{y}))$. Po trochu zdĺhavejších algebraických úpravách dostávame bilineárnu formu

$$\begin{aligned} F^{-1}(F(\mathbf{x}) \cdot F(\mathbf{y})) &= \frac{1}{4} \begin{bmatrix} \mu_{0,0}(\mathbf{x}) \\ \mu_{1,0}(\mathbf{x}) \\ \mu_{0,1}(\mathbf{x}) \\ \mu_{1,1}(\mathbf{x}) \end{bmatrix}^{\mathrm{T}} \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & -1 & 1 & -1 \\ 1 & 1 & -1 & -1 \\ 1 & -1 & -1 & 1 \end{bmatrix} \begin{bmatrix} \mu_{0,0}(\mathbf{y}) \\ \mu_{1,0}(\mathbf{y}) \\ \mu_{0,1}(\mathbf{y}) \\ \mu_{1,1}(\mathbf{y}) \end{bmatrix} = \\ &= \frac{1}{4} \begin{bmatrix} \mu_{0,0}(\mathbf{x}) + \mu_{1,1}(\mathbf{x}) \\ \mu_{1,0}(\mathbf{x}) - \mu_{0,1}(\mathbf{x}) \\ \mu_{0,0}(\mathbf{x}) - \mu_{1,1}(\mathbf{x}) \\ \mu_{1,0}(\mathbf{x}) + \mu_{0,1}(\mathbf{x}) \end{bmatrix}^{\mathrm{T}} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \mu_{0,0}(\mathbf{y}) + \mu_{1,1}(\mathbf{y}) \\ \mu_{0,1}(\mathbf{y}) - \mu_{1,0}(\mathbf{y}) \\ \mu_{1,0}(\mathbf{y}) + \mu_{0,1}(\mathbf{y}) \\ \mu_{0,0}(\mathbf{y}) - \mu_{1,1}(\mathbf{y}) \end{bmatrix}, \end{aligned}$$

[1] Frekvenčnou oblasťou sa nazýva množina všetkých možných výsledkov DFT, prípadne n-rozmernej DFT. U diskrétnej cas-cas transformácie toto označenie nie je celkom prirodzené, ale aj napriek tomu ho použijeme.

FAST-REAL-CIRCULAR-2D-CONVOLUTION (N_1, N_2: `integer`, $\mathbf{x}, \mathbf{y}$: `matrix of real`)

```
1   variables
2      k1, k2: integer
3      x1, x2, x3, x4, y1, y2, y3, y4: real
4      Z, z: matrix of real
5   X ← FCCT(N1, N2, x)
6   Y ← FCCT(N1, N2, y)
7   for k1 ← 0 to N1 − 1 do                          ▷ výpočet F^{-1}(F(X) · F(Y))
8      for k2 ← 0 to N2 − 1 do begin
9         x1 ← X[k1, k2]
10        x2 ← X[(−k1) mod N1, k2]
11        x3 ← X[k1, (−k2) mod N1]
12        x4 ← X[(−k1) mod N1, (−k2) mod N1]
13        y1 ← Y[k1, k2]
14        y2 ← Y[(−k1) mod N1, k2]
15        y3 ← Y[k1, (−k2) mod N1]
16        y4 ← Y[(−k1) mod N1, (−k2) mod N1]
17        Z[k1, k2] ← 0,25 · ((x1 + x4) · (y1 + y4) + (x2 − x3) · (y3 − y2) +
18                            +(x1 − x4) · (y2 + y3) + (x2 + x3) · (y1 − y4))
19     end
20  z ← FCCT^{-1}(N1, N2, Z)
21  return z · √(N1 N2)
```

Obr. 2.3: Rýchly algoritmus výpočtu dvojrozmernej diskrétnej cyklickej konvolúcie použitím cas-cas transformácie. Hodnoty N_1, N_2 vyjadrujú rozmery matíc $\mathbf{x}, \mathbf{y}$.

ktorú možno prepísať do jednoduchého tvaru

$$F^{-1}(F(\mathbf{x}) \cdot F(\mathbf{y})) = \tfrac{1}{4}[(\mathbf{x} + \mu_{1,1}(\mathbf{x}))(\mathbf{y} + \mu_{1,1}(\mathbf{y})) + (\mu_{1,0}(\mathbf{x}) - \mu_{0,1}(\mathbf{x}))(\mu_{0,1}(\mathbf{y}) - \mu_{1,0}(\mathbf{y})) + (\mathbf{x} - \mu_{1,1}(\mathbf{x}))(\mu_{1,0}(\mathbf{y}) + \mu_{0,1}(\mathbf{y})) + (\mu_{1,0}(\mathbf{x}) + \mu_{0,1}(\mathbf{x}))(\mathbf{y} - \mu_{1,1}(\mathbf{y}))] .$$

Keďže diskrétna cas-cas transformácia je k sebe inverzná, je na riadku č. 20 tohto algoritmu možné nahradiť volanie FCCT^{-1} volaním doprednej cas-cas transformácie, FCCT.

Porovnajme teraz časovú zložitosť algoritmu na výpočet dvojrozmernej diskrétnej cyklickej konvolúcie používajúceho rýchlu dvojrozmernú Fourierovu transformáciu (obr. 2.1) so zložitosťou algoritmu používajúceho rýchlu cas-cas transformáciu (obr. 2.3). Označme časovú zložitosť algoritmu založeného na Fourierovej transformácii ako $T_1(n)$ a časovú zložitosť algoritmu založeného na cas-cas transformácii ako $T_2(n)$. Je zrejmé, že obe zložitosti možno vyjadriť nasledovne:

$$T_1(n) = a_1 n^2 \lg n + b_1 n^2 + o(n^2) ,$$
$$T_2(n) = a_2 n^2 \lg n + b_2 n^2 + o(n^2) ,$$

kde člen a_1 závisí iba od efektívnosti implementácie algoritmu na výpočet diskrétnej Fourierovej transformácie a a_2 závisí od implementácie algoritmu na výpočet diskrétnej cas-cas transformácie. Členy b_1 a b_2 zrejme závisia navyše aj od kódu na riadkoch 6 až 8 algoritmu FAST-CIRCULAR-2D--CONVOLUTION na obr. 2.1, resp. 7 až 19 algoritmu FAST-REAL-CIRCULAR-2D-CONVOLUTION na obr. 2.3. Keďže algoritmus FCCT implementovaný pre reálne 2D postupnosti vykonáva polovičný počet aritmetických operácií v porovnaní s algoritmom 2D FFT, je koeficient a_1 väčší ako koeficient a_2, a teda konvolučný algoritmus založený na FCCT je asymptoticky rýchlejší ako algoritmus založený na FFT. Otázkou však zostáva, pre aké veľké 2D postupnosti začína byť rýchlejší v skutočnosti. Odpoveď nie je jednoznačná, závisí totiž od konkrétnej implementácie týchto algoritmov.

Kapitola 3

Teória rozmiestňovania geometrických útvarov v diskrétnej rovine

V tejto kapitole uvedieme definíciu diskrétnej roviny, geometrických útvarov, ich vzájomných polôh a definíciu rozmiestnenia geometrických útvarov.

3.1. Minkowského súčet

Minkowského súčet je vo veľkej miere používaný napr. pri plánovaní pohybu robotov a pri analýze obrazu [25]. Použitím Minkowského súčtu a diferencie možno pretransformovať problém zistenia prieniku, resp. inklúzie geometrických útvarov na jednoduchší test, či daný bod leží vnútri alebo mimo geometrického útvaru.

Definícia 3.1 (Minkowského súčet, doplnok, posunutie, symetrický obraz).
Nech $(G,+)$ je abelovská grupa a M, N sú podmnožiny G. *Minkowského súčet* množín M, N je definovaný (viď napr. [9], [10], [19], [25]):

$$M \oplus N = \{\, \mathbf{a}+\mathbf{b} \mid \mathbf{a} \in M, \mathbf{b} \in N \,\} \ .$$

Ďalej definujeme *doplnok* množiny, *posunutie* množiny (pre $\mathbf{t} \in G$) a *symetrický obraz* množiny:

$$\begin{aligned} \overline{M} &= \{\, \mathbf{a} \in G \mid \mathbf{a} \notin M \,\} = G \setminus M \ , \\ M+\mathbf{t} &= \{\, \mathbf{a}+\mathbf{t} \mid \mathbf{a} \in M \,\} = M \oplus \{\mathbf{t}\} \ , \\ -M &= \{\, -\mathbf{a} \mid \mathbf{a} \in M \,\} \ . \end{aligned}$$

Budeme hovoriť, že množina M je *symetrická*, ak $M = -M$.

Lema 3.2 (Niektoré vlastnosti Minkowského súčtu).
Minkowského súčet je komutatívny a asociatívny:

$$\begin{aligned} M \oplus N &= N \oplus M \ , \\ (M \oplus N) \oplus O &= M \oplus (N \oplus O) \ , \end{aligned}$$

invariantný voči posunutiu a preklopeniu:

$$(M+\mathbf{s})\oplus(N+\mathbf{t}) = (M\oplus N)+(\mathbf{s}+\mathbf{t})\ ,$$
$$-(M\oplus N) = (-M)\oplus(-N)\ ,$$

a je ho možné vyjadriť ako zjednotenie posunutých množín:

$$M\oplus N = \bigcup_{\mathbf{t}\in N}(M+\mathbf{t})\ .$$

Dôkaz. Uvedené vlastnosti možno dokázať priamo z definície Minkowského súčtu. ■

Lema 3.3.
Nech $(G,+)$ je abelovská grupa, M, N sú podmnožiny G a $\mathbf{t}\in G$. Potom $(M+\mathbf{t})\cap N\neq\emptyset$ vtedy a len vtedy, keď $\mathbf{t}\in N\oplus(-M)$.

Dôkaz.
„⇒“: Nech $\mathbf{n}\in(M+\mathbf{t})\cap N$. Teda $\mathbf{n}\in(M+\mathbf{t})$ a $\mathbf{n}\in N$. T.j. existuje $\mathbf{m}\in M$ také, že $\mathbf{n}=\mathbf{m}+$ $+\mathbf{t}$, resp. $\mathbf{t}=\mathbf{n}+(-\mathbf{m})$. Ak si všimneme, že $\mathbf{n}+(-\mathbf{m})$ je bod z $N\oplus(-M)$, dostávame priamo $\mathbf{t}\in N\oplus(-M)$.
„⇐“: Ak $\mathbf{t}\in N\oplus(-M)$, potom máme $\mathbf{t}=\mathbf{n}+(-\mathbf{m})$, pre nejaké $\mathbf{m}\in M$ a $\mathbf{n}\in N$. Ak prepíšeme $\mathbf{t}=\mathbf{n}+(-\mathbf{m})$ na $\mathbf{n}=\mathbf{m}+\mathbf{t}$, môžeme si všimnúť, že $n\in N$ a $\mathbf{n}\in(M+\mathbf{t})$, z čoho vyplýva $(M+\mathbf{t})\cap N\neq\emptyset$. ■

Dôsledok 3.4.
Nech $(G,+)$ je abelovská grupa, M, N sú podmnožiny G a $\mathbf{s},\mathbf{t}\in G$. Potom $(M+\mathbf{s})\cap(N+\mathbf{t})\neq\emptyset$ práve vtedy, keď $(\mathbf{s}-\mathbf{t})\in N\oplus(-M)$.

Dôsledok 3.5.
Množiny M a N majú neprázdny prienik práve vtedy, keď množina $N\oplus(-M)$ obsahuje neutrálny prvok grupy G.

3.2. Geometrické útvary v diskrétnej rovine

Definícia 3.6 (Diskrétna rovina, susednosť bodov).
Diskrétna rovina je množina $\mathbb{Z}^2$ s binárnou operáciou $+$ a reláciou $\mathcal{N}$, pričom $(\mathbb{Z}^2,+)$ je abelovská grupa, kde binárna operácia $+$ je definovaná nasledovne:

$$[a_1,a_2]+[b_1,b_2] = [a_1+b_1,a_2+b_2]\ ,$$

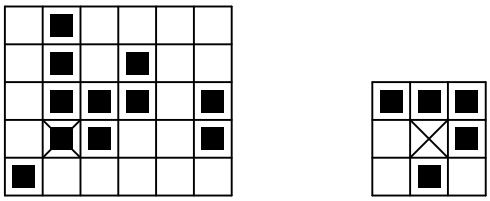
čiže dvojici *bodov* (prvkov množiny $\mathbb{Z}^2$, ďalej tiež *vektorov*) priradí bod, ktorý vznikne ich sčítaním po zložkách, a $\mathcal{N}$ (*susednosť bodov*) je binárna relácia na množine $\mathbb{Z}^2$, ktorá je

1. antireflexívna (žiadny bod nesusedí sám so sebou),
2. symetrická (ak bod **a** susedí s bodom **b**, tak aj bod **b** susedí s bodom **a**) a
3. invariantná voči posunutiu (susednosť dvoch bodov nezávisí od ich absolútnej pozície).

Hovoríme, že dva body **a**, **b** sú *susedné*, ak $\mathbf{a}\,\mathcal{N}\,\mathbf{b}$, inak hovoríme, že nie sú susedné. Susednosť $\mathcal{N}$ nazveme *konečnou*, ak bod $\mathbf{0} = [0,0]$ susedí s konečným počtom bodov.

Poznámka.
Dôležité je si uvedomiť, že uvedená definícia umožňuje modelovať aj diskrétnu rovinu s hexagonálnou topológiou[1].

Definícia 3.7 (Geometrický útvar v diskrétnej rovine).
Geometrický útvar v diskrétnej rovine $(\mathbb{Z}^2, +, \mathcal{N})$ (ďalej geometrický útvar) je každá množina $M \subseteq \mathbb{Z}^2$.



Obr. 3.1: Príklady geometrických útvarov. Body diskrétnej roviny prislúchajúce objektom sú označené čiernymi štvorčekmi. Počiatok súradnicovej sústavy je vyznačený krížikom.

Definícia 3.8 (Charakteristická funkcia geometrického útvaru).
Nech M je geometrický útvar v diskrétnej rovine $\mathbb{Z}^2$. *Charakteristická funkcia*[2] $M : \mathbb{Z}^2 \to \{0,1\}$ geometrického útvaru M je definovaná:

$$M(\mathbf{x}) = \begin{cases} 1, & \text{ak } \mathbf{x} \in M\,, \\ 0, & \text{ak } \mathbf{x} \notin M\,. \end{cases}$$

Lema 3.9.
Nech M a N sú geometrické útvary. Potom pre všetky $\mathbf{x} \in \mathbb{Z}^2$ platí:

$$(M \cap N)(\mathbf{x}) = M(\mathbf{x}) \cdot N(\mathbf{x})\,,$$
$$(M \cup N)(\mathbf{x}) = M(\mathbf{x}) + N(\mathbf{x}) - (M \cap N)(\mathbf{x})\,.$$

Definícia 3.10 (Obsah geometrického útvaru, ohraničený geom. útvar).
Obsah geometrického útvaru M budeme definovať ako mohutnosť tejto množiny, čiže $|M| \in \mathbb{N}_0 \cup \{\aleph_0\}$.
Ak $|M| \prec \aleph_0$, t.j. ak množina M je konečná, geometrický útvar M nazveme *ohraničený*, inak ho nazveme *neohraničený*.

Poznámka.
Ak M je ohraničený geometrický útvar, jeho obsah je rovný hodnote sumy $\sum_{\mathbf{x}\in\mathbb{Z}^2} M(\mathbf{x})$.

[1] Každý bod má práve šiestich susedov.

[2] Charakteristická funkcia sa zvyčajne označuje symbolom χ, my však taký zápis nebudeme používať, z dôvodu maximálnej jednoduchosti a prehľadnosti.

Definícia 3.11 (Obdĺžnik).
Geometrický útvar M sa nazýva *obdĺžnik*, ak preň existujú celé čísla $x_1 \leq x_2$ a $y_1 \leq y_2$ také, že pre všetky $[x,y] \in \mathbb{Z}^2$ platí[3]:

$$M[x,y] = \begin{cases} 1, & \text{ak } x_1 \leq x < x_2 \ \wedge \ y_1 \leq y < y_2 \ , \\ 0, & \text{inak} \ . \end{cases}$$

Definícia 3.12 (Obálka geometrického útvaru).
Nech M je geometrický utvar. Potom *obálkou* útvaru M, ak existuje, nazveme najmenší (z hľadiska inklúzie) obdĺžnik $\mathrm{Env}(M)$, pre ktorý platí $M \subseteq \mathrm{Env}(M)$.

Definícia 3.13 (Štruktúra susednosti).
Ľubovoľnú symetrickú množinu $S \subset \mathbb{Z}^2$ neobsahujúcu neutrálny prvok $\mathbf{0}$ grupy $(\mathbb{Z}^2, +)$ nazveme *štruktúra susednosti*.

Definícia 3.14 (S-susednosť bodov).
Nech S je nejaká štruktúra susednosti. Dva body $\mathbf{a}$, $\mathbf{b}$ sú *S-susedné* práve vtedy, keď $(\mathbf{a}-\mathbf{b}) \in S$ (poprípade $(\mathbf{b}-\mathbf{a}) \in S$, keďže S je podľa predpokladu symetrická).

Veta 3.15 (O existencii a jednoznačnosti štruktúry susednosti).
Nech $(\mathbb{Z}^2, +, \mathcal{N})$ je diskrétna rovina. Potom existuje práve jedna štruktúra susednosti $S_\mathcal{N}$ taká, že dva body $\mathbf{a}$, $\mathbf{b}$ sú susedné práve vtedy, keď sú $S_\mathcal{N}$-susedné.

Dôkaz. Položme $S_\mathcal{N} = \left\{ \mathbf{x} \in \mathbb{Z}^2 \;\middle|\; \mathbf{0}\,\mathcal{N}\,\mathbf{x} \right\}$. Najprv musíme ukázať, že $S_\mathcal{N}$ je štruktúra susednosti, t.j. že $S_\mathcal{N}$ je symetrická a neobsahuje bod $\mathbf{0}$.

- Symetrickosť $S_\mathcal{N}$ sa dá ukázať na základe symetrickosti a invariantnosti relácie $\mathcal{N}$ voči posunutiu. Nech teda $\mathbf{a} \in S_\mathcal{N}$. Ukážeme, že aj $-\mathbf{a} \in S_\mathcal{N}$. Keďže $\mathbf{a} \in S_\mathcal{N}$, máme $\mathbf{0}\,\mathcal{N}\,\mathbf{a}$. Z invariantnosti $\mathcal{N}$ voči posunutiu vyplýva $(\mathbf{0}-\mathbf{a})\,\mathcal{N}\,(\mathbf{a}-\mathbf{a})$, t.j. $(-\mathbf{a})\,\mathcal{N}\,\mathbf{0}$. Zo symetrie $\mathcal{N}$ zasa dostávame $\mathbf{0}\,\mathcal{N}\,(-\mathbf{a})$, čo v konečnom dôsledku znamená, že $-\mathbf{a} \in S_\mathcal{N}$.
- Keďže $\mathcal{N}$ je antireflexívna relácia, platí $\neg(\mathbf{0}\,\mathcal{N}\,\mathbf{0})$, čo znamená, že $\mathbf{0} \notin S_\mathcal{N}$.

Teraz dokážeme ekvivalenciu susednosti a $S_\mathcal{N}$-susednosti bodov $\mathbf{a}$ a $\mathbf{b}$:

$$\begin{aligned} (\mathbf{a}-\mathbf{b}) \in S_\mathcal{N} \Leftrightarrow (\mathbf{a}-\mathbf{b}) \in \left\{ \mathbf{x} \in \mathbb{Z}^2 \;\middle|\; \mathbf{0}\,\mathcal{N}\,\mathbf{x} \right\} \Leftrightarrow \mathbf{a} \in \left\{ \mathbf{x}' \in \mathbb{Z}^2 \;\middle|\; \mathbf{0}\,\mathcal{N}\,(\mathbf{x}'-\mathbf{b}) \right\} \Leftrightarrow \\ \Leftrightarrow \mathbf{a} \in \left\{ \mathbf{x}' \in \mathbb{Z}^2 \;\middle|\; \mathbf{b}\,\mathcal{N}\,\mathbf{x}' \right\} \Leftrightarrow \mathbf{b}\,\mathcal{N}\,\mathbf{a} \Leftrightarrow \mathbf{a}\,\mathcal{N}\,\mathbf{b} \ . \end{aligned}$$

Týmto sme dokončili prvú časť dôkazu, a to existenciu štruktúry susednosti. Zostáva už len dokázať, že je jediná. Nech by teda existovali dve navzájom rôzne štruktúry susednosti S_1 a S_2 také, že $\mathbf{a}\,\mathcal{N}\,\mathbf{b} \Leftrightarrow (\mathbf{a},\mathbf{b}$ sú S_1-susedné$)$ a zároveň $\mathbf{a}\,\mathcal{N}\,\mathbf{b} \Leftrightarrow (\mathbf{a},\mathbf{b}$ sú S_2-susedné$)$. Z tohto zrejme vyplýva, že $\mathbf{a}$, $\mathbf{b}$ sú S_1-susedné práve vtedy, keď sú S_2-susedné. Teda $(\mathbf{a}-\mathbf{b}) \in S_1$ práve vtedy, keď $(\mathbf{a}-\mathbf{b}) \in S_2$, čo je zrejme spor s predpokladom existencie dvoch navzájom rôznych štruktúr susednosti s uvedenou vlastnosťou. ■

[3] Namiesto $M([x,y])$ píšeme pre jednoduchosť $M[x,y]$.

Definícia 3.16.
Nech $(\mathbb{Z}^2,+,\mathcal{N})$ je diskrétna rovina. O štruktúre susednosti $S_{\mathcal{N}} = \left\{ \mathbf{x} \in \mathbb{Z}^2 \;\middle|\; \mathbf{0}\,\mathcal{N}\,\mathbf{x} \right\}$ z predchádzajúcej vety budeme hovoriť, že *prislúcha* diskrétnej rovine $(\mathbb{Z}^2,+,\mathcal{N})$.

Poznámka.
V diskrétnej rovine sa zvyknú používať dva druhy susednosti: S_4-susednosť a S_8-susednosť (niekedy nazývané ako 4-susednosť, resp. 8-susednosť), ktorých štruktúry susednosti sú definované nasledovne:

$$S_4 = \{[1,0],[0,1],[-1,0],[0,-1]\},$$
$$S_8 = \{[1,0],[1,1],[0,1],[-1,1],[-1,0],[-1,-1],[0,-1],[1,-1]\}.$$

Grafické znázornenie týchto štruktúr je na Obr. 3.2.

Obr. 3.2: Štruktúry susednosti S_4 a S_8

3.2.1. Vzájomné polohy geometrických útvarov

Definícia 3.17 (Prekryv a styk geometrických útvarov).
Nech M, N sú geometrické útvary v diskrétnej rovine $\mathbb{Z}^2$:
Budeme hovoriť, že:

- M a N sa *prekrývajú*, ak $M \cap N \neq \emptyset$,
- M a N sa *stýkajú* (tiež sú *husto rozmiestnené*), ak sa neprekrývajú a existujú dva také body $\mathbf{a} \in M$ a $\mathbf{b} \in N$ také, že $\mathbf{a}$ a $\mathbf{b}$ sú susedné.

Lema 3.18.
Nech M a N sú geometrické útvary v diskrétnej rovine $(\mathbb{Z}^2,+,\mathcal{N})$ s príslušnou štruktúrou susednosti $S_{\mathcal{N}}$. Potom M a N sa stýkajú práve vtedy, keď $M \cap N = \emptyset$ a $(M \oplus S_{\mathcal{N}}) \cap N \neq \emptyset$.

Definícia 3.19 (Rozmiestnenie geometrických útvarov v oblasti).
Nech $M_1, M_2, \ldots, M_n$ sú geometrické útvary a M_0 je *oblasť* (tiež geometrický útvar) v $\mathbb{Z}^2$. Hovoríme, že útvary $M_1, M_2, \ldots, M_n$ sú *rozmiestnené* v oblasti M_0 ak spĺňajú dve nasledujúce podmienky:

- $\forall i \in \{1,\ldots,n\} : M_i \subseteq M_0$,
- Ak $i \neq j$, potom M_i a M_j sa neprekrývajú.

Definícia 3.20 (Husté rozmiestnenie geometrických útvarov).
Geometrické útvary $M_1, M_2, \ldots, M_n$ sú *husto rozmiestnené*, ak sa dajú tak usporiadať, že každý z nich je husto rozmiestnený so zjednotením predchádzajúcich.

Definícia 3.21 (Konfiguračný priestor).
Pre všetky geometrické útvary M, N označme:

$$\mathrm{O}(M,N) = \left\{ \mathbf{t} \in \mathbb{Z}^2 \;\middle|\; M+\mathbf{t} \text{ a } N \text{ sa prekrývajú} \right\} ,$$
$$\mathrm{D}(M,N) = \left\{ \mathbf{t} \in \mathbb{Z}^2 \;\middle|\; M+\mathbf{t} \text{ a } N \text{ sa stýkajú} \right\} .$$

Doplnok množiny $\mathrm{O}(M,N)$ sa nazýva *konfiguračný priestor pre umiestnenie* geometrického útvaru M vzľadom na útvar N.

Veta 3.22.
Pre akékoľvek dva geometrické útvary M, N platí:

$$\mathrm{O}(M,N) = N \oplus -M ,$$
$$\mathrm{D}(M,N) = ((N \oplus -M) \oplus -S_{\mathcal{N}}) \setminus (N \oplus -M) ,$$

kde $S_{\mathcal{N}}$ je štruktúra susednosti diskrétnej roviny.

Dôkaz. Prvá rovnosť je priamym dôsledkom lemy 3.3. Druhá rovnosť vyplýva z definície 3.17, z lemy 3.2, z lemy 3.18 a z prvej rovnosti.

$$\begin{aligned}
\mathrm{D}(M,N) &= \left\{ \mathbf{t} \in \mathbb{Z}^2 \;\middle|\; M+\mathbf{t} \text{ a } N \text{ sa stýkajú} \right\} = \\
&= \left\{ \mathbf{t} \in \mathbb{Z}^2 \;\middle|\; (M+\mathbf{t}) \oplus S_{\mathcal{N}} \text{ a } N \text{ sa prekrývajú, ale } M+\mathbf{t} \text{ a } N \text{ sa neprekrývajú} \right\} = \\
&= \mathrm{O}(M \oplus S_{\mathcal{N}}, N) \setminus \mathrm{O}(M,N) = (N \oplus -(M \oplus S_{\mathcal{N}})) \setminus (N \oplus -M) = \\
&= ((N \oplus -M) \oplus -S_{\mathcal{N}}) \setminus (N \oplus -M) .
\end{aligned}$$

■

Definícia 3.23 (Translačná konfigurácia, ohodnotenie konfigurácie).
Nech $M_1, \ldots, M_n$ sú geometrické útvary v diskrétnej rovine $\mathbb{Z}^2$. Usporiadanú n-ticu $[\mathbf{x}_1, \ldots, \mathbf{x}_n]$ vektorov zo $\mathbb{Z}^2$ nazveme *(translačnou) konfiguráciou* útvarov $M_1, \ldots, M_n$ v rovine $\mathbb{Z}^2$. Ľubovoľnú funkciu $\mathrm{Val}_{M_1,\ldots,M_n}$ zobrazujúcu n-ticu $[\mathbf{x}_1, \ldots, \mathbf{x}_n]$ na reálne číslo (prípadne na prvok nejakej inej lineárne usporiadanej množiny), ktorá spĺňa nasledujúcu podmienku: existuje taká konštanta c, že pre všetky konfigurácie $[\mathbf{x}_1, \ldots, \mathbf{x}_n]$ platí

$$\mathrm{Val}_{M_1,\ldots,M_n}[\mathbf{x}_1, \ldots, \mathbf{x}_n] \geq c, \text{ ak } M_1+\mathbf{x}_1, \ldots, M_n+\mathbf{x}_n \text{ sú rozmiestnené v } \mathbb{Z}^2 ,$$
$$\mathrm{Val}_{M_1,\ldots,M_n}[\mathbf{x}_1, \ldots, \mathbf{x}_n] < c, \text{ inak,}$$

nazveme *ohodnotením konfigurácie*[4], pričom hovoríme, že konfigurácia A je *lepšia* ako konfigurácia B, ak

$$\mathrm{Val}_{M_1,\ldots,M_n}(A) > \mathrm{Val}_{M_1,\ldots,M_n}(B) .$$

Konfigurácie, ktorých ohodnotenie je väčšie alebo rovné konštante c nazývame *platnými konfiguráciami*,[5] ostatné nazývame *neplatnými konfiguráciami*.

[4] V prípade potreby je možné zameniť symboly „$\geq$“ za „$>$“ a „$<$“ za „$\leq$“ a následne pozmeniť definíciu platných konfigurácií.

[5] Rozmiestnenia útvarov sú formované práve platnými konfiguráciami.

Poznámka.
Je ľahké si predstaviť viacero príkladov ohodnotení konfigurácií:

- **Triviálne ohodnotenie**, ktoré priraďuje hodnotu 1 platným konfiguráciam a hodnotu 0 neplatným konfiguráciam. Jeho využitie je pravdepodobne čisto teoretické, pretože zrejme nie je vhodné pre žiaden prakticky použiteľný optimalizačný model.
- **Ohodnotenie založené na efektívnosti využitia plochy**. Jedným z takýchto ohodnotení môže byť napríklad také, ktoré priraďuje platným konfiguráciam $[\mathbf{x}_1, \ldots, \mathbf{x}_n]$ útvarov $M_1, \ldots, M_n$ hodnotu

$$\sum_{k=1}^{n} |M_k| \; / \; \left| \mathrm{Env}\left(\bigcup_{k=1}^{n} M_k + \mathbf{x}_k \right) \right| ,$$

  t.j. mieru využitia plochy obálky zjednotenia útvarov. Neplatným konfiguráciam by takéto ohodnotenie malo priraďovať hodnotu 0, prípadne nejakým spôsobom penalizovať „horšie“ konfigurácie (s väčším prekryvom útvarov).

Existujú aj iné typy ohodnotení, napríklad ohodnotenia zohľadňujúce spôsob „uloženia“ útvarov. Bližšie sa konkrétnymi príkladmi budeme zaoberať neskôr.

Lema 3.24.
Nech $M_1, \ldots, M_n$ sú geometrické útvary a M_0 je oblasť v $\mathbb{Z}^2$. Útvary $M_1 + \mathbf{x}_1, \ldots, M_n + \mathbf{x}_n$ sú rozmiestnené v oblasti M_0 práve vtedy, keď $[\mathbf{0}, \mathbf{x}_1, \ldots, \mathbf{x}_n]$ je platná konfigurácia útvarov $\overline{M_0}, M_1, \ldots, M_n$.

Definícia 3.25 (Translačný kontajnment, optimalizačný problém).
Problém translačného kontajnmentu v diskrétnej rovine možno špecifikovať nasledujúcim spôsobom (viď problém kNN pre euklidovskú rovinu v [10]):
Nech $M_1, \ldots, M_n$ sú geometrické útvary a M_0 je oblasť v $\mathbb{Z}^2$. Úlohou je nájsť (ak existuje) takú translačnú konfiguráciu $[\mathbf{x}_1, \ldots, \mathbf{x}_n]$ týchto útvarov, aby útvary $(M_1 + \mathbf{x}_1), \ldots, (M_n + \mathbf{x}_n)$ boli rozmiestnené v oblasti M_0, teda aby

$$\forall i \in \{1, \ldots, n\} : (M_i + \mathbf{x}_i) \subseteq M_0 ,$$
$$\forall i, j \in \{1, \ldots, n\} : i \neq j \Rightarrow \mathbf{x}_i - \mathbf{x}_j \in \overline{\mathrm{O}(M_i, M_j)} .$$

Optimalizačný problém translačného kontajnmentu je rozšírený navyše o podmienku, aby ohodnotenie nájdenej konfigurácie bolo maximálne, t.j. aby nájdená konfigurácia bola v istom zmysle najlepšia. Často sa ako oblasť M_0 používa jedným smerom nekonečný pás tvaru obdĺžnika. Ohodnotenie v takomto prípade zohľadňuje veľkosť obálky útvarov $M_1 + \mathbf{x}_1, \ldots, M_n + \mathbf{x}_n$.

Poznámka.
Ak oblasť, v ktorej rozmiestňujeme útvary, je konvexná[6], potom optimálna konfigurácia týchto útvarov formuje husté rozmiestnenie.

[6]Pojem konvexnosti útvarov v diskrétnej rovine zatiaľ nedefinujeme. V prípade 4-susednosti by sme mohli povedať, že útvar je konvexný, ak je obdĺžnikového tvaru, v prípade 8-susednosti, ak je osemuholníkového tvaru.

3.3. Zrýchlený algoritmus výpočtu obsahov prienikov útvarov v diskrétnej rovine

Ukážeme, že existuje efektívny algoritmus, ktorý pre dané dva geometrické útvary v diskrétnej rovine zostaví tabuľku, v ktorej sa pre každé vzájomné posunutie týchto dvoch útvarov, keď sa pretínajú ich obálky, bude nachádzať veľkosť (obsah, mohutnosť) ich prieniku. Výhodou vytvorenia takejto tabuľky je, že na výpočet obsahu prieniku dvoch posunutých útvarov, pre ktoré bola skonštruovaná takáto tabuľka, je potrebný iba konštantný čas. V tabuľke stačí mať uložené hodnoty len pre posunutia týchto útvarov, pri ktorých sa pretínajú ich obálky. Je totiž zrejmé, že ak sa obálky útvarov nepretínajú, nemôžu sa pretínať ani samotné útvary.

Všimnime si teraz obsah prieniku geometrického útvaru M posunutého o vektor $\mathbf{t}$ s útvarom N (použijeme lemu 3.9):

$$\begin{aligned} |(M+\mathbf{t})\cap N| &= \sum_{\mathbf{a}\in\mathbb{Z}^2} ((M+\mathbf{t})\cap N)(\mathbf{a}) = \sum_{\mathbf{a}\in\mathbb{Z}^2} (M+\mathbf{t})(\mathbf{a})\cdot N(\mathbf{a}) = \\ &= \sum_{\mathbf{a}\in\mathbb{Z}^2} M(\mathbf{a}-\mathbf{t})\cdot N(\mathbf{a}) = \sum_{\mathbf{a}\in\mathbb{Z}^2} N(\mathbf{a})\cdot(-M)(\mathbf{t}-\mathbf{a})\,. \end{aligned} \tag{3.1}$$

Posledná suma nápadne pripomína definíciu (dvojrozmernej) diskrétnej konvolúcie. Konkrétne, ak položíme (viď definíciu 3.8)

$$\begin{aligned} \mathbf{a} &= [k_1,k_2]\,, & \mathbf{t} &= [n_1,n_2]\,, \\ \mathbf{x}_{k_1,k_2} &= N(\mathbf{a})\,, & \mathbf{y}_{n_1-k_1,n_2-k_2} &= (-M)(\mathbf{t}-\mathbf{a})\,, \end{aligned}$$

potom táto suma prejde do tvaru

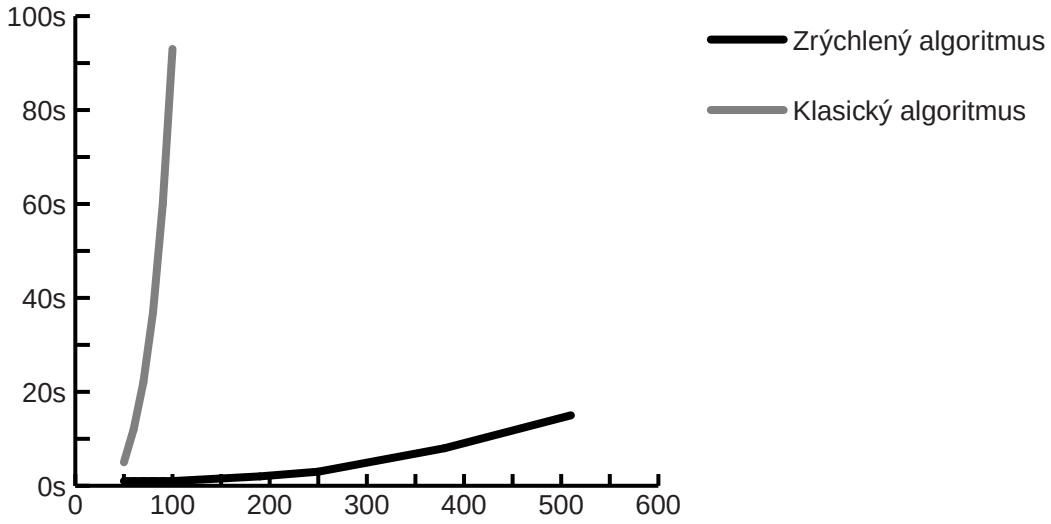
$$\sum_{\mathbf{a}\in\mathbb{Z}^2} N(\mathbf{a})\cdot(-M)(\mathbf{t}-\mathbf{a}) = \sum_{k_1\in\mathbb{Z}}\sum_{k_2\in\mathbb{Z}} \mathbf{x}_{k_1,k_2}\mathbf{y}_{n_1-k_1,n_2-k_2}\,, \tag{3.2}$$

čo je zrejme suma z definície 2.7 dvojrozmernej diskrétnej konvolúcie. Ak predpokladáme, že geometrické útvary M a N sú ohraničené, potom postupnosti $\mathbf{x}$ a $\mathbf{y}$ budú zrejme nulové mimo nejakých obdĺžnikov, veľkosti $M_r\times M_s$, resp. $N_r\times N_s$. Na základe vety 2.9 je výsledkom operácie konvolúcie 3.2 dvojrozmerná postupnosť nulová mimo obdĺžnika veľkosti $m\times n$, kde $m=M_r+{}+N_r-1$ a $n=M_s+N_s-1$ a jej hodnoty je možné priamo vypočítať pomocou dvojrozmernej cyklickej konvolúcie matíc pozostávajúcich z hodnôt postupností $\mathbf{x}$ a $\mathbf{y}$ doplnených nulami na veľkosť $m\times n$. Čas potrebný na výpočet dvojrozmernej cyklickej konvolúcie matíc tejto veľkosti je $O(mn\lg mn)$. Ak naviac oba geometrické útvary majú veľkosť rádovo $n\times n$, teda $m=O(n)$, potom tento čas je $O(n^2\lg n)$. V prípade, že by sme poslednú sumu zo vzťahu 3.1 počítali pre každý vektor $\mathbf{t}$ osamote, takto vzniknutý algoritmus by mal zrejme časovú zložitosť $\Theta(n^4)$, ktorá sa pohybuje už na hranici praktickej použiteľnosti. Porovnanie časových zložitostí zrýchleného algoritmu a algoritmu vychádzajúceho z definícií sa nachádza na obr. 3.4.

COMPUTE-AREAS (M, N: `2D shape`)

```
1  variables
2     M, N, A: integer 2D sequence
3  M ← −M
4  Zakóduj geometrické útvary M a N ako celočíselné 2D postupnosti M, N.
5  A ← FAST-2D-CONVOLUTION(N, M)
6  return A
```

Obr. 3.3: Rýchly algoritmus výpočtu obsahov prienikov útvarov v diskrétnej rovine. Pri použití tohto algoritmu predpokladáme, že aspoň jeden zo vstupných útvarov je ohraničený (resp. oba, ak použijeme procedúru na výpočet konvolúcie, ktorá vyžaduje v podstate nulové 2D postupnosti). Ak je táto podmienka splnená, potom bude vrátená postupnosť **A**, ktorá pre každé možné posunutie **t** bude obsahovať obsah prieniku útvaru $M + \mathbf{t}$ s útvarom N. Postupnosť **A** môže byť (v prípade, že je v podstate nulová) kódovaná ako matica spolu s vektorom jej umiestnenia v diskrétnej rovine.



Obr. 3.4: Orientačné porovnanie rýchlosti behu zrýchleného algoritmu a klasického algoritmu na výpočet matice obsahov prienikov útvarov vychádzajúceho z definícií. Čísla na vodorovnej osi vyjadrujú veľkosť útvarov, t.j. napríklad číslo 400 znamená, že procedúra bola vykonaná pre dva útvary veľkosti 400×400. Horizontálna os vyjadruje čas behu algoritmov. Meranie bolo uskutočnené na počítači s procesorom Intel® Pentium II™ 266MHz. Zrýchlený algoritmus využíval algoritmus FFT-2D (pre reálne matice) implementovaný v knižnici FFTW (viď [14]).

3.4. Zrýchlený algoritmus výpočtu Minkowského súčtov v diskrétnej rovine

V [25] možno nájsť zložitosti niektorých algoritmov na výpočet Minkowského súčtov mnohouholníkov v euklidovskej rovine. My pracujeme v diskrétnej rovine, preto zvolíme iný prístup, a to využitím rýchleho algoritmu výpočtu cyklickej konvolúcie, ktorý bol odvodený vyššie.

Priamym dôsledkom vety 3.22 a komutatívnosti Minkowského súčtu je skutočnosť, že pre ľubovoľné dva geometrické útvary M a N platí rovnosť $M \oplus N = O(-M, N)$.

$$O(-M,N) = \left\{ \mathbf{t} \in \mathbb{Z}^2 \;\middle|\; -M+\mathbf{t} \text{ a } N \text{ sa prekrývajú} \right\} = \\ = \left\{ \mathbf{t} \in \mathbb{Z}^2 \;\middle|\; |(-M+\mathbf{t}) \cap N| \succ 0 \right\} ,$$

zrejme teda $(M \oplus N)(\mathbf{t}) = \Upsilon|(-M+\mathbf{t}) \cap N|$, kde $\Upsilon(x)$ je jednotková Heavisidova funkcia[7]. Na základe výsledkov predchádzajúceho odseku sme schopní vypočítať charakteristickú funkciu Minkowského súčtu dvoch geometrických útvarov (v diskrétnej rovine) podobným spôsobom:

$$(M \oplus N)(\mathbf{t}) = \Upsilon \left(\sum_{\mathbf{a} \in \mathbb{Z}^2} N(\mathbf{a}) M(\mathbf{t}-\mathbf{a}) \right) .$$

COMPUTE-MINKOWSKI-SUM (M, N: `2D shape`)

```
1  variables
2     A: integer 2D sequence
3     A: 2D shape
4  A ← COMPUTE-AREAS(−M, N)
5  Na základe hodnôt postupnosti A vytvor geometrický útvar A (bod bude patriť
6  útvaru A práve vtedy, keď hodnota postupnosti A v tomto bode bude rôzna od 0).
7  return A
```

Obr. 3.5: Rýchly algoritmus výpočtu Minkowského súčtov geometrických útvarov v diskrétnej rovine. Pre jednoduchosť sme využili algoritmus COMPUTE-AREAS, takže útvar M sa teraz zbytočne dvakrát po sebe preklápa podľa bodu **0**.

Časová zložitosť tohto algoritmu je asymptoticky rovná zložitosti algoritmu COMPUTE--AREAS, teda $O(n^2 \lg n)$, ak oba geometrické útvary majú veľkosť rádovo $n \times n$.

3.5. Ďalšie optimalizácie

3.5.1. Optimalizácia pamäťových prístupov

Ak na výpočet Minkowského súčtov, resp. tabuliek obsahov prienikov dvoch útvarov chceme aplikovať uvedený zrýchlený algoritmus využívajúci 2D-DFT alebo DCCT, narazíme na problém optimalizácie pamäťových prístupov. O algoritmoch FFT (aj pre vektorové a paralelné počítače)

[7] Jednotková Heavisidova funkcia (jednotková skoková funkcia) je definovaná predpisom:

$$\Upsilon(x) = \begin{cases} 1, & \text{ak } x > 0 , \\ 0, & \text{inak.} \end{cases}$$

je totiž známe, že sú nevhodné pre spracovanie veľkého objemu dát uložených či už v externej alebo v hierarchickej pamäti (viď [2]).

Jedným z riešení je napríklad problém modifikovať na vyššej úrovni, t.j. namiesto jedného výpočtu konvolúcie rozsiahlych matíc, tieto matice rozdeliť na niekoľko menších a výslednú konvolúciu zložiť z čiastkových konvolúcií („každého s každým“, bipartitne) týchto matíc menších rozmerov. Týmto sa však badateľne zvýši časová zložitosť, i keď iba o konštantný faktor. Ak totiž oba útvary rozdelíme na 4 rovnako veľké časti (z hľadiska rozmerov obálok), časová zložitosť narastie približne štvornásobne. Bližšie informácie týkajúce sa výpočtu konvolúcie (iba jednorozmernej) možno nájsť v [35].

Druhým riešením je využiť napríklad tzv. algoritmus FFT v štyroch krokoch (Four Step Algorithm) alebo iné techniky uvedené v [1] a [2].

3.5.2. Presnosť zrýchleného algoritmu

Jednou z negatívnych vlastností tohto algoritmu je, že pracuje s reálnymi číslami, ktoré sa na žiadnom počítači nedajú reprezentovať s maximálnou presnosťou. Každé reálne číslo je nahrádzané nejakou aproximáciou z konečnej množiny čísel, čím pri výpočtoch vzniká kvantizačný šum. Priamym dôsledkom tohto šumu je v lepšom prípade pokles presnosti výsledkov. Teda uvedený algoritmus, v prípade, že konvolúcia bude realizovaná pomocou DFT alebo DHYT, nie je celkom presný a môže poskytovať chybné výsledky[8].

Tento nedostatok je možné odstrániť tak, že algoritmus dvojrozmernej celočíselnej konvolúcie bude využívať iný typ transformácie (viď [1], [17], [27]), konkrétne Fourierovu transformáciu na konečnom poli (napr. v modulárnej aritmetike na celých číslach). V tejto oblasti však zatiaľ zostáva viacero nevyriešených otázok.

[8] Vzniknuté chyby však možno eliminovať zaokrúhľovaním.

Kapitola 4

Stochastické metódy rozmiestňovania

Vzhľadom na veľkú zložitosť problému automatického rozmiestňovania geometrických útvarov nie je prakticky možné systematicky testovať všetky konfigurácie útvarov, aby sme mohli vybrať tú najlepšiu, preto sa ako alternatívne riešenie používajú rôzne približné a stochastické algoritmy (viď napr. [5], [10], [11], [30]). Ak chceme pomocou týchto metód riešiť problém optimálneho rozmiestnenia útvarov, musíme najprv popísať postup, ktorým možno porovnať dve konfigurácie geometrických útvarov. Preto sa najprv zameriame na jednu z možných definícií ohodnotenia translačných konfigurácií, ktorej maximum budeme za pomoci týchto algoritmov hľadať. Doteraz publikované metódy porovnávania konfigurácií vychádzali zvyčajne z efektivity využitia plochy oblasti, do ktorej boli útvary rozmiestňované. Prezentovaná metóda prináša niekoľko výhod, napríklad má také vlastnosti, ktoré umožňujú rýchlejšiu konvergenciu optimalizačných algoritmov. Ďalšou výhodou je napríklad možnosť efektívneho výpočtu ohodnotenia konfigurácie (ak konfigurácia pozostáva z n útvarov, v prípade, že vôbec nezohľadníme topologické vlastnosti konfigurácií, na výpočet ohodnotenia budeme potrebovať čas iba $O(n^2)$). Ešte poznamenajme, že budeme definovať iba ohodnotenie vhodné na hľadanie rozmiestnení útvarov v oblasti $\mathbb{Z}^2$. Relatívne jednoduchým spôsobom ho bude možné rozšíriť pre rozmiestňovanie útvarov v iných oblastiach ako $\mathbb{Z}^2$.

4.1. Ohodnotenie translačných konfigurácií

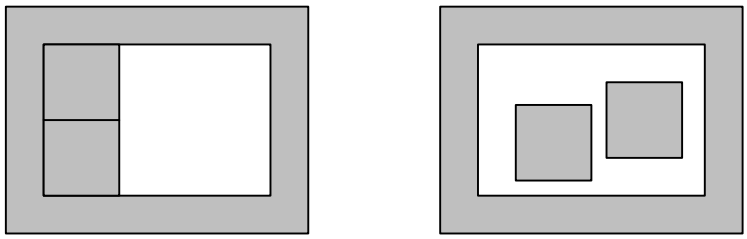
Majme n geometrických útvarov $M_1,\ldots,M_n$ v diskrétnej rovine $\mathbb{Z}^2$. Chceme nájsť také ohodnotenie konfigurácií $[\mathbf{x}_1,\ldots,\mathbf{x}_n]$ týchto útvarov, ktorého graf by obsahoval minimum plošín (súvislých oblastí s konštantnou hodnotou) a ktorého hodnoty by boli tým väčšie, čím je menšia plocha obálky útvarov $M_1+\mathbf{x}_1$ až $M_n+\mathbf{x}_n$. Dôvodom je to, že väčšina optimalizačných algoritmov (simulované žíhanie, genetické algoritmy, ...) pracuje lepšie, ak optimalizované funkcie obsahujú menej konštantných oblastí.

Zrejme ohodnotenie konfigurácie $[\mathbf{x}_1,\ldots,\mathbf{x}_n]$ útvarov $M_1,\ldots,M_n$, ktoré nadobúda väčšie hodnoty pre konfigurácie s menšou plochou obálky útvarov $M_1+\mathbf{x}_1$ až $M_n+\mathbf{x}_n$ je možné definovať

nasledujúcim spôsobom (podľa poznámky na strane 29):

$$\mu^{\text{eff}}_{M_1,\ldots,M_n}[\mathbf{x}_1,\ldots,\mathbf{x}_n] = \begin{cases} \sum_{k=1}^{n} |M_k| \;/\; \left|\text{Env}\left(\bigcup_{k=1}^{n} M_k + \mathbf{x}_k\right)\right| , & \text{ak } [\mathbf{x}_1,\ldots,\mathbf{x}_n] \text{ je platná,} \\ 0 , & \text{inak.} \end{cases}$$

Nevýhodou tohto ohodnotenia je skutočnosť, že jeho graf obsahuje množstvo oblastí s konštantnou hodnotou, čo má za následok, že optimalizačné algoritmy (najmä tie, čo sú založené na gradientových metódach) pri hľadaní (sub-)optimálneho riešenia zlyhávajú. Na obrázku 4.1 sú znázornené dve rôzne konfigurácie troch útvarov, ktoré majú rovnakú hodnotu μ^{eff}. Konfigurácia a) sa nám zdá byť úspornejšia v porovnaní s konfiguráciou b) lebo v obálke vytvára väčší súvislý priestor, ktorý je možné využiť pri vkladaní ďalších útvarov.



a) b)

Obr. 4.1: Obe znázornené konfigurácie (tých istých troch útvarov vyplnených šedou farbou) majú rovnaké ohodnotenie μ^{eff}, hoci konfigurácia a) sa aj laikovi zdá byť lepšia ako konfigurácia b). Obálky majú totiž rovnakú veľkosť a útvary sa neprekrývajú. Možno si všimnúť, že ohodnotenie je rovnaké pre všetky platné konfigurácie, ktoré majú rovnakú veľkosť obálky. Tento fakt predstavuje významnú prekážku pri hľadaní (sub-)optimálnej konfigurácie (z hľadiska ohodnotenia) pomocou klasických optimalizačných algoritmov.

Teraz budeme definovať ohodnotenie konfigurácií, ktoré bude nadobúdať väčšie hodnoty pre konfigurácie, v ktorých sú útvary na seba viacej „pritisnuté“, t.j. napríklad konfigurácia a) z obr. 4.1 bude lepšia (pri tomto ohodnotení) ako konfigurácia b) z toho istého obrázku.

Najprv si všimneme, akým spôsobom možno ohodnotiť konfigurácie dvoch geometrických útvarov tak, aby graf ohodnotenia (funkcie zo $\mathbb{Z}^4$ do $\mathbb{R}$) obsahoval minimum plošín. Následne uvedieme spôsob, ako možno túto definíciu rozšíriť pre ohodnotenie konfigurácie n útvarov.

Jednou z možných definícií ohodnotenia konfigurácie $[\mathbf{x}_1,\mathbf{x}_2]$ útvarov M_1 a M_2 je

$$\mu^{\text{tight.1}}_{M_1,M_2}[\mathbf{x}_1,\mathbf{x}_2] = \begin{cases} |((M_1+\mathbf{x}_1)\oplus E)\cap((M_2+\mathbf{x}_2)\oplus E)| , & \text{ak } [\mathbf{x}_1,\mathbf{x}_2] \text{ je platná,} \\ -|(M_1+\mathbf{x}_1)\cap(M_2+\mathbf{x}_2)| , & \text{inak,} \end{cases}$$

kde E je nejaká neprázdna konečná symetrická množina, najlepšie buď štruktúra susednosti roviny alebo nejaký obdĺžnik so stredom v bode $\mathbf{0}$. Výraz $|(M_1+\mathbf{x}_1)\cap(M_2+\mathbf{x}_2)|$ vyjadruje, nakoľko „zlá“ je daná konfigurácia, ak sa útvary prekrývajú. Nevýhodou ohodnotenia definovaného týmto spôsobom je, že hodnoty priraďované jednotlivým konfiguráciam príliš závisia od obsahov útvarov M_1 a M_2. Treba poznamenať, že táto funkcia iba hodnotí ako dobre sú dané dva

objekty k sebe priložené[1], intuitívne by sa dalo povedať, že meria niečo ako dĺžku spoločného obvodu (prieniku obvodov rozšírených množinou E), a teda vôbec nezohľadňuje veľkosť obálky zjednotenia týchto útvarov.

Ďalšou možnosťou je čiastočne normalizovať alebo aspoň ohraničiť hodnoty, ktoré môže hodnotiaca funkcia nadobúdať. Vezmime funkciu definovanú nasledovne (M_1, M_2 a E sú neprázdne množiny):

$$\mu^{\text{tight.2}}_{M_1,M_2}[\mathbf{x}_1,\mathbf{x}_2] = \begin{cases} \mu^{\text{tight.1}}_{M_1,M_2}[\mathbf{x}_1,\mathbf{x}_2] / \min\{|M_1 \oplus E|, |M_2 \oplus E|\}\,, & \text{ak } [\mathbf{x}_1,\mathbf{x}_2] \text{ je platná,} \\ \mu^{\text{tight.1}}_{M_1,M_2}[\mathbf{x}_1,\mathbf{x}_2] / \min\{|M_1|, |M_2|\}\,, & \text{inak.} \end{cases}$$

O tejto funkcii vieme, že nadobúda hodnoty z intervalu $\langle -1, 1\rangle$. Navyše vieme, že ak sa objekty $M_1 + \mathbf{x}_1$ a $M_2 + \mathbf{x}_2$ prekrývajú, nadobúda záporné hodnoty, a ak sa neprekrývajú nadobúda nezáporné hodnoty. To isté platí samozrejme aj o funkcii $\mu^{\text{tight.1}}$.

Teraz pristúpime k rozšíreniu poslednej definície na n útvarov. Od hodnotiacej funkcie požadujeme, aby funkčné hodnoty boli záporné pre neplatné konfigurácie a pre platné konfigurácie boli kladné, prípadne nulové. Ohodnotenie možno teraz definovať nasledujúcim spôsobom:

$$\mu^{\text{tight}}_{M_1,\ldots,M_n}[\mathbf{x}_1,\ldots,\mathbf{x}_n] = \sum_{1\le i<j\le n} \mu^{\text{tight.2}}_{M_i,M_j}[\mathbf{x}_i,\mathbf{x}_j] + \nu_{M_1,\ldots,M_n}[\mathbf{x}_1,\ldots,\mathbf{x}_n]\,,$$

$$\nu_{M_1,\ldots,M_n}[\mathbf{x}_1,\ldots,\mathbf{x}_n] = \begin{cases} 0 & \text{ak } [\mathbf{x}_1,\ldots,\mathbf{x}_n] \text{ je platná konfigurácia,} \\ -\frac{n(n-1)}{2} & \text{inak.} \end{cases}$$

Úlohou korekčného člena $\nu_{M_1,\ldots,M_n}[\mathbf{x}_1,\ldots,\mathbf{x}_n]$ je zabezpečiť, aby funkcia $\mu_{M_1,\ldots,M_n}[\mathbf{x}_1,\ldots,\mathbf{x}_n]$ bola ohodnotením. Jeho definícia využíva fakt, že hodnoty $\mu^{\text{tight.2}}_{M_i,M_j}[\mathbf{x}_i,\mathbf{x}_j]$ sú menšie alebo rovné 1.

Hoci graf posledného skonštruovaného ohodnotenia μ^{tight} obsahuje v porovnaní s ohodnotením μ^{eff} menej konštantných oblastí, má jednu závažnú negatívnu vlastnosť. Toto ohodnotenie nie je totiž skoro vôbec vhodné na minimalizáciu veľkosti zvyšnej plochy z obálky útvarov. Povedzme, že by množina E bola obdĺžnikom so stredom v bode $\mathbf{0}$. Ak by $E = S_8$ alebo $E = S_4$, ohodnotenie by bolo aspoň čiastočne použiteľné na minimalizáciu obsahu zvyšnej plochy, ale graf ohodnotenia by sa vyznačoval väčším množstvom plošín (hoci by to stále bolo lepšie ako v prípade μ^{eff}). Ak by E bol väčší obdĺžnik, graf by mal lepší tvar, ale toto ohodnotenie by skoro vôbec nebolo použiteľné na optimalizáciu využitia plochy obálky (bližšie viď príklad na obr. 4.2).

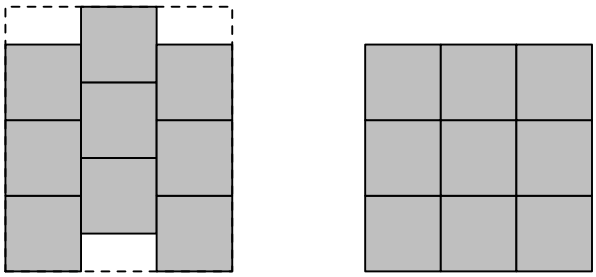
Teraz pristúpime ku definícii ohodnotenia, ktoré spája pozitívne vlastnosti oboch ohodnotení μ^{eff} a μ^{tight}. V podstate je ho možné skonštruovať nasledujúcim spôsobom:

$$\mu_{M_1,\ldots,M_n}[\mathbf{x}_1,\ldots,\mathbf{x}_n] = [\mu^{\text{eff}}_{M_1,\ldots,M_n}[\mathbf{x}_1,\ldots,\mathbf{x}_n], \mu^{\text{tight}}_{M_1,\ldots,M_n}[\mathbf{x}_1,\ldots,\mathbf{x}_n]]\,.$$

Rozdielom oproti predchádzajúcim definíciam ohodnotení je to, že μ je funkcia zobrazujúca konfigurácie útvarov do množiny $\mathbb{R}^2$ namiesto do $\mathbb{R}$, pričom na množine $\mathbb{R}^2$ definujeme (lexikografické) usporiadanie týmto spôsobom:

$$[a_1,a_2] < [b_1,b_2] \Leftrightarrow (a_1 < b_1) \vee ((a_1 = b_1) \wedge (a_2 < b_2))\,.$$

[1] Samozrejme iba vtedy, keď sa útvary neprekrývajú. Ak sa prekrývajú, hodnotou funkcie je záporný obsah ich prieniku.



Obr. 4.2: Ak množina E je väčší obdĺžnik so stredom v bode **0**, potom konfigurácia a) môže byť hodnotená funkciou μ^{tight} lepšie ako konfigurácia b). Dôvodom je skutočnosť, že ohodnotenie μ^{tight} vôbec nezohľadňuje efektívnosť využitia plochy obálky útvarov.

4.2. Jednoduché metódy automatického rozmiestňovania

V tejto časti sú popísané niektoré jednoduché metódy optimalizácie funkcií. Nebudeme detailne popisovať spôsob ich použitia na automatické rozmiestňovanie geometrických útvarov. Principiálne je možné rozmiestňovať útvary nasledujúcim spôsobom. Vyberieme si niektorú z uvedených metód a pomocou nej sa budeme snažiť nájsť takú konfiguráciu geometrických útvarov, ktorej ohodnotenie, napr. μ, je maximálne.

4.2.1. Náhodné prehľadávanie

Najjednoduchším prístupom ku riešeniu hľadania optima zložitých funkcií je pravdepodobne náhodné alebo enumerované prehľadávanie. Body prehľadávaného priestoru sú volené náhodne, prípadne nejakým systematickým spôsobom, pričom sú ohodnocované. Nie je to príliš inteligentná stratégia a zriedkakedy sa používa samostatne.

Naivný algoritmus hľadajúci optimálnu konfiguráciu z hľadiska ohodnotenia μ sa nachádza na obr. 4.3. Tento algoritmus postupne generuje náhodné konfigurácie, pričom po vopred určenom počte iterácií vráti nájdenú konfiguráciu s maximálnym ohodnotením μ.

4.2.2. Gradientové metódy

Na optimalizáciu spojitých funkcií bolo vyvinutých množstvo rozličných metód využívajúcich informácie o gradiente optimalizovanej funkcie[2]. Na základe tohto gradientu je určovaný smer ďalšieho prehľadávania. Ak však nie je možné vypočítať alebo odhadnúť deriváciu vyšetrovanej funkcie, napríklad pretože nie je spojitá, tieto metódy zvyčajne zlyhávajú.

[2]Hoci gradientové metódy nepatria medzi stochastické metódy, zohrávajú významnú úlohu napríklad pri iterovanom náhodnom prehľadávaní alebo pri memetických algoritmoch.

```
RANDOM-SEARCH (n: integer, M: array of 2D shape)
1  variables
2     x, y: array of vector from ℤ²
3     xe, xt, ye, yt: extended real
4  ye, yt ← −∞
5  for i ← 1 to MaxIterations do
6     Vygeneruj náhodnú konfiguráciu x.
7     xe ← μ^eff_{M1,...,Mn}[x1, ..., xn]
8     if xe > ye
9        y ← x
10    else
11       if xe = ye
12          xt ← μ^tight_{M1,...,Mn}[x1, ..., xn]
13          if xt > yt
14             y ← x
15 return y
```

Obr. 4.3: Naivný algoritmus hľadania optimálnej konfigurácie. Algoritmus postupne generuje náhodné konfigurácie, pričom po vopred určenom počte iterácií (*MaxIterations*) vráti tú konfiguráciu, ktorej ohodnotenie μ je maximálne.

Gradientové metódy pracujú vynikajúco na funkciách s jediným maximom, resp. minimom, ale na funkciách s viacerými lokálnymi maximami, resp. minimami[3] zlyhávajú z dôvodu, že nenájdu globálne optimum, ale skončia v prvom nájdenom lokálnom optime.

V prípade optimalizácie funkcií s diskrétnym definičným oborom môžu tieto metódy pracovať na základe informácií o diskrétnom gradiente.

4.2.3. Iterované prehľadávanie

Náhodné a gradientové prehľadávanie je možné skombinovať do tzv. iterovaného prehľadávania. Pri tejto metóde sa opakovane vykonáva nejaký gradientový algoritmus. V okamihu, keď je nájdené lokálne optimum, zvolí sa náhodne nejaký bod prehľadávaného priestoru a proces gradientového prehľadávania sa opakuje. Výhodou tejto metódy je jej jednoduchosť a to, že s veľkou pravdepodobnosťou nájde globálne optimum, ak funkcia nemá príliš veľa lokálnych optím.

4.2.4. Simulované žíhanie

Táto metóda je v podstate modifikovanou gradientovou metódou. Prehľadávanie sa začína v náhodnom bode priestoru. Následne sa uskutoční náhodný posun. Ak nás tento posun povedie ku bodu s väčšou hodnotou, akceptujeme ho. Ak nás povedie ku bodu s menšou hodnotou, prijmeme ho s pravdepodobnosťou $p(t)$, kde t je čas. Funkcia $p(t)$ začína s hodnotou blízkou 1,

[3] Takéto funkcie sa nazývajú multimodálne.

ale jej hodnoty sa postupne znižujú smerom k nule. Je to proces analogický ku ochladzovaniu roztaveného materiálu.

Spočiatku sú akceptované všetky posuny, ale zároveň so znižovaním „teploty“ klesá aj pravdepodobnosť prijatia posunu smerom k nižším hodnotám. Negatívne posuny sú potrebné na únik od lokálnych maxím, ale priveľa takýchto posunov nás vedie ďalej od globálneho maxima.

Podobne ako náhodné prehľadávanie, aj simulované žíhanie narába v danom okamihu iba s jedným potenciálnym riešením, takže si nevytvára celkový obraz o prehľadávanom priestore. Neukladajú sa žiadne informácie o predchádzajúcich posunoch, na základe ktorých by bolo možné riadiť výber nasledujúcich posunov (tento nedostatok čiastočne odstraňuje tzv. tabu prehľadávanie, viď [23]).

4.3. Genetické algoritmy

Genetické algoritmy sú triedou veľmi silných optimalizačných metód založených na paradigme biologickej evolúcie. Existuje veľké množstvo problémov, ktoré sa dajú efektívne riešiť pomocou týchto metód, ale na druhej strane, existujú aj problémy, na ktorých genetické algoritmy úplne zlyhávajú. Hlavnou výhodou genetických algoritmov je ich schopnosť paralelného prehľadávania a schopnosť kombinovať prehľadávanie priestoru ako do šírky, tak aj do hĺbky. Genetické algoritmy sú bližšie popísané v dodatku B.

Problematika automatického rozmiestňovania geometrických útvarov použitím genetických algoritmov bola rozoberaná v [5], kde boli dosiahnuté celkom zaujímavé výsledky. Populácia jedincov bola tvorená viacerými rozmiestneniami útvarov v euklidovskej rovine, ktoré boli reprezentované pomocou hierarchických štruktúr[4]. Jednotlivé útvary boli kódované pomocou tzv. hrebeňového kódu (angl. comb-code), ktorý pre daný útvar jednoduchým spôsobom popisoval plochu medzi obrysom útvaru a jeho obdĺžnikovou obálkou. Dobrou vlastnosťou tohto kódu bola možnosť rýchleho výpočtu približnej efektívnosti využitia plochy daným rozmiestnením útvarov.

4.3.1. Návrh systému na automatické rozmiestňovanie útvarov

Najvýhodnejším typom stochastických algoritmov hľadajúcich (sub-)optimálne rozmiestnenia geometrických útvarov sú pravdepodobne genetické algoritmy. Dôvodom je, už uvedená, schopnosť paralelného prehľadávania a schopnosť kombinovaného prehľadávania priestoru do šírky a do hĺbky. Takisto, výhodou použitia genetických algoritmov je možnosť definovať spôsob kódovania útvarov a ich konfigurácií, pričom pri definícii môžeme zohľadňovať aj topologické charakteristiky týchto konfigurácií, ktoré sú dôležité najmä pri modifikácii rozmiestnení, prípadne pri konštrukcii nového rozmiestnenia na základe iných, dobre hodnotených, rozmiestnení.

Systém bude mať k dispozícii pre každú dvojicu útvarov M_1, M_2 predpočítané dve tabuľky, v ktorých pre každé možné vzájomné posunutie $\mathbf{x}$ útvarov, pri ktorom sa pretínajú ich obálky, budú uložené obsahy prienikov $(M_1 + \mathbf{x}) \cap M_2$ a $((M_1 \oplus E) + \mathbf{x}) \cap (M_2 \oplus E)$. Pomocou týchto

[4] V uvedenej práci bola použitá stromová reprezentácia rozmiestnení. Listami stromu boli jednotlivé útvary, ktoré boli zoskupované do väčších útvarov vnútornými uzlami reprezentujúcimi napríklad priloženie zľava alebo sprava. Takisto bola zohľadnená aj možnosť rotácie útvarov o násobky 90°.

tabuliek je takto možné veľmi efektívne (z hľadiska časovej zložitosti) v priebehu optimalizácie počítať ohodnotenia translačných konfigurácií μ^{tight}, prípadne je možné navyše predpočítať ohodnotenia $\mu^{\text{tight.2}}$. Výpočet μ^{tight} danej konfigurácie n-útvarov sa dá vykonať v čase $O(n^2)$, ktorý je pravdepodobne možné ešte zredukovať, ak sa položia určité obmedzenia na charakter útvarov, ktoré následne umožnia využiť topologické vlastnosti rozmiestnení.

Poznamenajme ešte, že navrhovaný systém nebude umožňovať žiadne rotácie útvarov, hoci v diskrétnej rovine so susednosťou S_4 alebo S_8, by nemal nastať väčší problém. Pamäťová zložitosť môže narásť približne štvornásobne, časová zložitosť zrejme tiež porastie, keďže pre každý rozmiestňovaný útvar pridávame ďalší stupeň voľnosti.

Dôležité je uvedomiť si, že pri praktických aplikáciach budú často rozmiestňované množiny útvarov, v ktorých sa môžu vyskytovať útvary rovnakého tvaru. Zvyčajne sa totiž z pásu látky vyrezávajú dielce pre viacero kusov odevu naraz, aby sa ešte výraznejšie šetrilo materiálom. Z tohto dôvodu je možné ešte znížiť pamäťovú zložitosť, niekedy až 100-násobne[5].

Teraz približne popíšeme použitie genetických algoritmov v automatickom systéme rozmiestňovania geometrických útvarov v oblasti $\mathbb{Z}^2$.

Chromozómy

Chromozómy reprezentujú potenciálne riešenia – konfigurácie geometrických útvarov. Budú pozostávať z dvoch spájaných zoznamov. Jeden bude obsahovať všetky útvary[6], ktoré už sú umiestnené v rovine, pričom pre každý útvar v ňom bude uložená pozícia tohto útvaru. Tento zoznam označíme ako $\mathbf{L}_1$. Druhý zoznam, $\mathbf{L}_2$, bude pozostávať z útvarov, ktoré nie sú umiestnené do roviny, ale pre každý z nich v tomto zozname bude zaznamenaná pozícia jeho posledného umiestnenia, ak bol niekedy umiestnený do roviny. Každý chromozóm teda predstavuje nejakú konfiguráciu útvarov úplne popisovanú údajmi v prvom zozname. Informácie v druhom zozname majú len pomocnú úlohu. Fakt, že sa útvar nachádza v druhom zozname, môžeme chápať buď tak, že ešte nie je umiestnený do roviny, alebo už je v rovine, ale na „veľmi zlej“ pozícii.

V tejto fáze, keď sme už definovali, čo je to chromozóm, je nutné povedať, kedy je jeden chromozóm lepší ako druhý chromozóm. Chromozóm bude tým lepší, čím viacej útvarov sa nachádza v prvom zozname, teda čím viacej útvarov z danej množiny je umiestnených do roviny. Ak majú dva chromozómy v prvých zoznamoch rovnaký počet útvarov, lepší bude ten, ktorý má väčšiu hodnotu z hľadiska funkcie μ. Konkrétne, ak prvý zoznam chromozómu A obsahuje útvary $M_1, \ldots, M_n$ umiestnených na pozíciach $\mathbf{x}_1, \ldots, \mathbf{x}_n$ a prvý zoznam chromozómu B obsahuje $N_1, \ldots, N_n$ na pozíciach $\mathbf{y}_1, \ldots, \mathbf{y}_n$, potom chromozóm A je lepší ako B práve vtedy, keď

$$\mu_{M_1,\ldots,M_n}[\mathbf{x}_1, \ldots, \mathbf{x}_n] < \mu_{N_1,\ldots,N_n}[\mathbf{y}_1, \ldots, \mathbf{y}_n] \,.$$

Fitness funkcia bude teda v tomto prípade zobrazovať priestor chromozómov (potenciálnych riešení) do množiny $\mathbb{N} \times \mathbb{R} \times \mathbb{R}$, na ktorej je definované lexikografické usporiadanie.

[5] Napríklad vtedy, keď odev bude pozostávať z 20 navzájom rôznych dielcov a algoritmus bude rozmiestňovať až 200 útvarov (pre 10 kusov odevu naraz).

[6] V zozname nebudú uložené bitové mapy, ale iba identifikátory útvarov.

Počiatočná populácia

Aby sa mohol začať proces optimalizácie, treba najprv definovať chromozómy v počiatočnej populácii. Jedným z možných riešení je začínať s populáciou, v ktorej všetky chromozómy majú prázdne prvé zoznamy. Počiatočná populácia by nemala mať príliš veľa prvkov, najlepší počet bude pravdepodobne v rozsahu 10 až 100 chromozómov (viď napr. [5], [20], [38]).

Operátory

Dôležitým operátorom definovaným nad priestorom chromozómov je *mutácia*. Mutácia je lokálny operátor, ktorého úlohou je zabezpečiť prehľadanie oblastí priestoru reprezentovaných chromozómami. Operátor mutácie pre chromozómy definované vyššie môže vykonávať nasledujúce zmeny na chromozóme:

1. Zo zoznamu $\mathbf{L}_1$ vyberie náhodný útvar a presunie ho na náhodnú pozíciu.
2. Zo zoznamu $\mathbf{L}_1$ vyberie náhodný útvar a presunie ho o náhodný vektor, ale nie príliš ďaleko od jeho pôvodnej pozície.
3. Zo zoznamu $\mathbf{L}_1$ vyberie náhodné dva útvary a zamení ich pozície.
4. Zvolí náhodný obdĺžnik, ktorý má neprázdny prienik s obdĺžnikovou obálkou zjednotenia útvarov prvého zoznamu a všetky útvary, ktoré sú podmnožinou tohto obdĺžnika (prípadne s ním majú neprázdny prienik) presunie o náhodný vektor.
5. Z náhodne vybraného zoznamu presunie náhodne zvolený útvar do druhého (nie vždy $\mathbf{L}_2$) zoznamu, t.j. presúva útvar buď z $\mathbf{L}_1$ do $\mathbf{L}_2$ alebo naopak.

Uvedené zmeny treba vykonávať náhodne, s určitou pravdepodobnosťou, nie deterministicky. Ďalej treba poznamenať, že je lepšie manipulovať skôr s menšími útvarmi ako s väčšími, teda je výhodné do operátora mutácie zakomponovať to, že menšie útvary budú selektované s väčšou pravdepodobnosťou ako útvary väčšie. Výnimkou môže byť snáď posledná uvedená možnosť (presun medzi zoznamami), pri ktorej zohľadňovanie veľkosti útvarov nie je veľmi žiadúce. Presunutie väčšieho útvaru do druhého zoznamu môže vytvoriť priestor pre umiestnenie ešte väčšieho útvaru.

Ďalším významným operátorom je operátor *rekombinácie* alebo *kríženia*, ktorého úlohou je prehľadávanie priestoru chromozómov do šírky. Operátor *kríženia* vezme dva chromozómy[7] (vybrané selekčným operátorom, viď dodatok B), na ktorých môže realizovať napríklad nasledujúcu operáciu.

Pre obe konfigurácie sa náhodne zvolia dve rovnobežné priamky, jedna pretínajúca obálku prvej konfigurácie a druhá pretínajúca obálku druhej konfigurácie útvarov. Obe množiny útvarov konfigurácií sa rozdelia na dve disjunktné podmnožiny, pričom prvá z podmnožín bude pozostávať z útvarov v jednej polrovine určenej priamkou a druhá z útvarov v druhej polrovine[8]. Útvary,

[7] Tieto chromozómy reprezentujú konfigurácie útvarov.

[8] Pre diskrétnu rovinu nedefinujeme pojem polroviny, ale vychádzame z predstavy polroviny v euklidovskej rovine, do ktorej je možné vnoriť diskrétnu rovinu. Podobne je to aj s pojmom priamky.

ktoré majú prienik s uvedenou priamkou, je vhodné zaradiť všetky spolu do jednej z podmnožín (pre každú konfiguráciu oddelene). Pre obe konfigurácie následne navzájom vymeníme podmnožiny útvarov prislúchajúce polrovinám na rovnakej strane určujúcich priamok. Pri výmene sa môžeme snažiť minimalizovať prieniky útvarov spolu s minimalizovaním veľkosti obálky útvarov konfigurácie. Vo väčšine prípadov môže nastať problém, že niektoré útvary (momentálne ide o tvar, nie o identitu útvarov) sa v konfiguráciach budú vyskytovať viacej krát, ako je povolené. Pre tieto útvary treba ešte uskutočniť finálne zmeny konfigurácií a odstrániť nadbytočnosť, prípadne deficienciu konfigurácií úpravami v zoznamoch chromozómov.

4.3.2. Ďalšie zrýchlenie

Pre každú dvojicu útvarov M_1, M_2 môžeme nájsť lokálne optimá ohodnotenia $\mu^{\text{tight.2}}_{M_1,M_2}$ a vybrať z nich tie, ktoré patria množine $\mathrm{D}(M_1, M_2)$. Každé takto nájdené posunutie reprezentuje „lokálne optimálne priloženie“ útvaru M_1 ku M_2. Teraz uvedieme postup tvorby množiny týchto posunutí:

1. Vypočítame maticu, v ktorej pre každú pozíciu $\mathbf{x}$ útvaru M_1 vzhľadom na M_2 bude uložená hodnota funkcie $\mu^{\text{tight.2}}_{M_1,M_2}[\mathbf{x}, \mathbf{0}]$.

2. Vo vzniknutej matici vyhľadáme všetky kladné lokálne maximá, ktoré patria množine $\mathrm{D}(M_1, M_2)$.

Takto vzniknutá množina lokálne optimálnych posunutí by mohla byť pre väčšie útvary (rádovo 1000×1000 bodov) zbytočne veľká, takže je možné zredukovať jej veľkosť napríklad nasledujúcimi spôsobmi:

- Lokálne maximá rozdelíme na $S_{\mathcal{N}}$-súvislé množiny rovnako ohodnotených bodov. Tieto množiny zrejme nie sú $S_{\mathcal{N}}$-susedné. Z každej takto vzniknutej množiny vyberieme niekoľko reprezentantov (najlepšie počet zhora ohraničený nejakou konštantou).

- V matici najprv vynulujeme všetky prvky, ktoré nie sú lokálnymi maximami. Potom budeme postupne prechádzať všetky prvky matice. Ak nájdeme nenulový prvok, zaradíme ho do zoznamu a všetky ostatné s rovnakou hodnotou v okne veľkosti napr. 8×8 vynulujeme[9].

Predpočítanie lokálne optimálnych vzájomných posunutí útvarov prináša niekoľko výhod. Je totiž viac pravdepodobné, že algoritmus bude konvergovať k optimálnemu riešeniu, ak počas výpočtu budeme s väčšou pravdepodobnosťou uskutočňovať lokálne optimálne výbery[10]. Napríklad operátor mutácie je možné pozmeniť tak, že s väčšou pravdepodobnosťou bude presúvať náhodne vybraný útvar na náhodne vybranú lokálne optimálnu pozíciu ako na náhodnú pozíciu. Takisto je možné upraviť aj ostatné prípady.

[9] Pri prechádzaní matice zľava doprava a zhora nadol stačí zrejme nulovať iba dolnú polovicu okna.

[10] Takýto algoritmus by sa dal nazvať stochastickým greedy algoritmom.

4.4. Memetické algoritmy

Memetické algoritmy[11] sú metódou heuristického prehľadávania založenou na paradigme kultúrnej evolúcie. Ukázalo sa, že v niektorých prípadoch konvergujú rádovo rýchlejšie ako tradičné genetické algoritmy. V princípe kombinujú heuristiky lokálneho prehľadávania s operátormi kríženia. Z tohto dôvodu ich niektorí odborníci považovali za hybridné genetické algoritmy, resp. genetické algoritmy s lokálnym prehľadávaním. Napriek tomu, do tejto triedy metaheuristík môžu patriť aj kombinácie s konštruktívnymi heuristikami alebo inými exaktnými metódami. Memetické algoritmy boli úspešne použité na vyriešnie viacerých rozsiahlych kombinatorických problémov, na ktorých ostatné metaheuristiky zlyhali.

Z uvedených dôvodov sa dá očakávať, že použitie memetických algoritmov môže ešte viac urýchliť konvergenciu systémov automatického rozmiestňovania geometrických útvarov.

[11] Prvý krát bol použitý pojem „memetický algoritmus“ r. 1989 v [31].

Dodatok A

Vzťah medzi 2D-DFT a cas-cas transformáciou

Cieľom tohto dodatku je ukázať, že transformácie 2D-DFT a cas-cas sú navzájom informačne zameniteľné, t.j. že jednu možno vypočítať z druhej a vice versa. Chceme teda dokázať že existuje také bijektívne zobrazenie F, ktoré spĺňa nasledujúce rovnosti pre všetky matice $\mathbf{x}$:

$$\Phi_{2\mathrm{D}}(\mathbf{x}) = F(\overline{h}_{2\mathrm{D}}(\mathbf{x})) \quad \text{a} \quad \overline{h}_{2\mathrm{D}}(\mathbf{x}) = F^{-1}(\Phi_{2\mathrm{D}}(\mathbf{x})) \,. \tag{A.1}$$

Pre zjednodušenie a skrátenie zápisu zavedieme nasledujúce označenia ($j \in \{1,2\}$):

$$C_j = \cos(2\pi k_j n_j / N_j) \,, \qquad S_j = \sin(2\pi k_j n_j / N_j) \,, \qquad r = \frac{1}{\sqrt{N_1 N_2}} \,.$$

Transformácie (2D-DFT a DCCT) môžeme po zavedení týchto označení písať:

$$\Phi_{2\mathrm{D}}(\mathbf{x})_{k_1,k_2} = r \sum_{n_1=0}^{N_1-1} \sum_{n_2=0}^{N_2-1} \mathbf{x}_{n_1,n_2} (C_1C_2 - \mathrm{i}S_1C_2 - \mathrm{i}C_1S_2 - S_1S_2) \,,$$
$$\overline{h}_{2\mathrm{D}}(\mathbf{x})_{k_1,k_2} = r \sum_{n_1=0}^{N_1-1} \sum_{n_2=0}^{N_2-1} \mathbf{x}_{n_1,n_2} (C_1C_2 + S_1C_2 + C_1S_2 + S_1S_2) \,.$$

Teraz ukážeme, že existuje také bijektívne zobrazenie F, ktoré spĺňa podmienku (A.1). Vyjadrime teraz všetky možné preklopenia matíc transformovaných pomocou DCCT:

$$\mu_{0,0}(\overline{h}_{2\mathrm{D}}(\mathbf{x}))_{k_1,k_2} = r \sum_{n_1=0}^{N_1-1} \sum_{n_2=0}^{N_2-1} \mathbf{x}_{n_1,n_2} (C_1C_2 + S_1C_2 + C_1S_2 + S_1S_2) \,,$$
$$\mu_{1,0}(\overline{h}_{2\mathrm{D}}(\mathbf{x}))_{k_1,k_2} = r \sum_{n_1=0}^{N_1-1} \sum_{n_2=0}^{N_2-1} \mathbf{x}_{n_1,n_2} (C_1C_2 - S_1C_2 + C_1S_2 - S_1S_2) \,,$$
$$\mu_{0,1}(\overline{h}_{2\mathrm{D}}(\mathbf{x}))_{k_1,k_2} = r \sum_{n_1=0}^{N_1-1} \sum_{n_2=0}^{N_2-1} \mathbf{x}_{n_1,n_2} (C_1C_2 + S_1C_2 - C_1S_2 - S_1S_2) \,,$$

$$\mu_{1,1}(\overline{h}_{2\mathrm{D}}(\mathbf{x}))_{k_1,k_2} = r\sum_{n_1=0}^{N_1-1}\sum_{n_2=0}^{N_2-1}\mathbf{x}_{n_1,n_2}(C_1C_2 - S_1C_2 - C_1S_2 + S_1S_2)\,,$$

a transformovaných použitím 2D-DFT:

$$\begin{aligned}
\mu_{0,0}(\Phi_{2\mathrm{D}}(\mathbf{x}))_{k_1,k_2} &= r\sum_{n_1=0}^{N_1-1}\sum_{n_2=0}^{N_2-1}\mathbf{x}_{n_1,n_2}(C_1C_2 - \mathrm{i}S_1C_2 - \mathrm{i}C_1S_2 - S_1S_2)\,,\\
\mu_{1,0}(\Phi_{2\mathrm{D}}(\mathbf{x}))_{k_1,k_2} &= r\sum_{n_1=0}^{N_1-1}\sum_{n_2=0}^{N_2-1}\mathbf{x}_{n_1,n_2}(C_1C_2 + \mathrm{i}S_1C_2 - \mathrm{i}C_1S_2 + S_1S_2)\,,\\
\mu_{0,1}(\Phi_{2\mathrm{D}}(\mathbf{x}))_{k_1,k_2} &= r\sum_{n_1=0}^{N_1-1}\sum_{n_2=0}^{N_2-1}\mathbf{x}_{n_1,n_2}(C_1C_2 - \mathrm{i}S_1C_2 + \mathrm{i}C_1S_2 + S_1S_2)\,,\\
\mu_{1,1}(\Phi_{2\mathrm{D}}(\mathbf{x}))_{k_1,k_2} &= r\sum_{n_1=0}^{N_1-1}\sum_{n_2=0}^{N_2-1}\mathbf{x}_{n_1,n_2}(C_1C_2 + \mathrm{i}S_1C_2 + \mathrm{i}C_1S_2 + S_1S_2)\,.
\end{aligned}$$

Hľadáme komplexné koeficienty $a_1,\ldots,a_4$ a $b_1,\ldots,b_4$ také, aby platilo

$$\begin{aligned}
\Phi_{2\mathrm{D}}(\mathbf{x}) &= a_1\mu_{0,0}(\overline{h}_{2\mathrm{D}}(\mathbf{x})) + a_2\mu_{1,0}(\overline{h}_{2\mathrm{D}}(\mathbf{x})) + a_3\mu_{0,1}(\overline{h}_{2\mathrm{D}}(\mathbf{x})) + a_4\mu_{1,1}(\overline{h}_{2\mathrm{D}}(\mathbf{x})) =\\
&= (a_1\mu_{0,0} + a_2\mu_{1,0} + a_3\mu_{0,1} + a_4\mu_{1,1})(\overline{h}_{2\mathrm{D}}(\mathbf{x}))\,,\\
\overline{h}_{2\mathrm{D}}(\mathbf{x}) &= b_1\mu_{0,0}(\Phi_{2\mathrm{D}}(\mathbf{x})) + b_2\mu_{1,0}(\Phi_{2\mathrm{D}}(\mathbf{x})) + b_3\mu_{0,1}(\Phi_{2\mathrm{D}}(\mathbf{x})) + b_4\mu_{1,1}(\Phi_{2\mathrm{D}}(\mathbf{x})) =\\
&= (b_1\mu_{0,0} + b_2\mu_{1,0} + b_3\mu_{0,1} + b_4\mu_{1,1})(\Phi_{2\mathrm{D}}(\mathbf{x}))\,.
\end{aligned}$$

Dostávame dve sústavy štyroch rovníc o štyroch neznámych

$$\begin{bmatrix} 1 & 1 & 1 & 1\\ 1 & -1 & 1 & -1\\ 1 & 1 & -1 & -1\\ 1 & -1 & -1 & 1 \end{bmatrix}\begin{bmatrix} a_1\\ a_2\\ a_3\\ a_4 \end{bmatrix} = \begin{bmatrix} 1\\ -\mathrm{i}\\ -\mathrm{i}\\ -1 \end{bmatrix}\,,\quad \begin{bmatrix} 1 & 1 & 1 & 1\\ -\mathrm{i} & \mathrm{i} & -\mathrm{i} & \mathrm{i}\\ -\mathrm{i} & -\mathrm{i} & \mathrm{i} & \mathrm{i}\\ -1 & 1 & 1 & -1 \end{bmatrix}\begin{bmatrix} b_1\\ b_2\\ b_3\\ b_4 \end{bmatrix} = \begin{bmatrix} 1\\ 1\\ 1\\ 1 \end{bmatrix}\,,$$

ktorých riešením sú:

$$\begin{aligned}
a_1 &= -i/2\,, \quad a_2 = 1/2\,, \quad a_3 = 1/2\,, \quad a_4 = \quad i/2\,.\\
b_1 &= \quad \mathrm{i}/2\,, \quad b_2 = 1/2\,, \quad b_3 = 1/2\,, \quad b_4 = -\mathrm{i}/2\,,
\end{aligned}$$

Teda našli sme priamy vzťah medzi DCCT a 2D-DFT, ktorý možno zapísať:

$$\begin{aligned}
\Phi_{2\mathrm{D}}(\mathbf{x}) &= \left(\frac{-\mathrm{i}\,\mu_{0,0} + \mu_{1,0} + \mu_{0,1} + \mathrm{i}\,\mu_{1,1}}{2}\right)(\overline{h}_{2\mathrm{D}}(\mathbf{x}))\,,\\
\overline{h}_{2\mathrm{D}}(\mathbf{x}) &= \left(\frac{\mathrm{i}\,\mu_{0,0} + \mu_{1,0} + \mu_{0,1} - \mathrm{i}\,\mu_{1,1}}{2}\right)(\Phi_{2\mathrm{D}}(\mathbf{x}))\,,
\end{aligned}$$

čo znamená, že hľadané zobrazenie F existuje a platí preň:

$$\begin{aligned} F(\mathbf{x}) &= \left(\frac{-\mathrm{i}\,\mu_{0,0} + \mu_{1,0} + \mu_{0,1} + \mathrm{i}\,\mu_{1,1}}{2} \right)(\mathbf{x}) \,, \\ F^{-1}(\mathbf{x}) &= \left(\frac{\mathrm{i}\,\mu_{0,0} + \mu_{1,0} + \mu_{0,1} - \mathrm{i}\,\mu_{1,1}}{2} \right)(\mathbf{x}) \,. \end{aligned} \tag{A.2}$$

Dodatok B

Genetické algoritmy

B.1. Biologická evolúcia

Genetické algoritmy tvoria triedu výpočtových modelov ktoré vznikli formalizáciou a následnou implementáciou jednej zo základných paradigiem živej prírody – *biologickej evolúcie*. Biologická evolúcia je progresívna zmena genetického obsahu populácie v priebehu veľkého počtu generácií. Pozostáva z nasledujúcich troch zložiek:

1. *Prirodzený výber* – proces, v ktorom jedinci s vysokou silou (*fitness*) vstupujú s väčšou pravdepodobnosťou do procesu reprodukcie ako jedinci s menšou silou.

2. *Náhodný genetický drift*, v ktorom náhodné udalosti (napr. náhodná mutácia genetického materiálu alebo náhodná smrť jedinca) v živote jedincov ovplyvňujú populáciu. Náhodné efekty genetického driftu zohrávajú významnú úlohu najmä pri malých populáciách.

3. *Reprodukčný proces*, v rámci ktorého sa z rodičov vytvárajú potomkovia. Genetická informácia potomkov je vytvorená kombináciou genetickej informácie rodičov. Zvyčajne tento proces prebieha tak, že z genetickej informácie dvoch jedincov vstupujúcich do procesu reprodukcie sa náhodne vyberú časti chromozómu, predstavujúce istú informáciu, na základe ktorých je potom zostavená genetická informácia nového jedinca – potomka. Tento proces sa tiež nazýva kríženie (sexuálna reprodukcia) a keďže sa vyskytuje u väčšiny zložitejších organizmov, je možné usudzovať, že podstatne zvyšuje rýchlosť a efektívnosť evolúcie.

V biológii je sila jedinca definovaná ako jeho relatívna schopnosť prežiť a reprodukovať sa v danom prostredí a danej populácii. V prírode, jedinci v populácii navzájom súťažia o zdroje akými sú napr. potrava alebo voda, poprípade, ak prislúchajú tomu istému druhu, súťažia o sexuálneho partnera. Tí jedinci, ktorí sú najúspešnejší z hľadiska schopnosti prežiť a získať sexuálneho partnera budú mať pravdepodobne väčší počet potomkov. Naopak, menej schopní jedinci vyprodukujú menší počet potomkov, poprípade nebudú mať žiadnych potomkov. To znamená, že genetická informácia dobre prispôsobených, silných jedincov sa s vysokou pravdepodobnosťou rozšíri na väčší počet indivíduí v nasledujúcej generácii. Kombináciou dobrých vlastností (z hľadiska sily) rôznych predkov môže niekedy vzniknúť veľmi silný jedinec. Týmto spôsobom sa druhy stále vyvíjajú a prispôsobujú svojmu prostrediu.

B.2. Genetické algoritmy

Základné princípy genetických algoritmov boli prvý krát uvedené v [20] a sú dobre popísané v mnohých textoch (napr. [3], [4], [15]). Genetické algoritmy sú stochastické optimalizačné algoritmy založené na princípe Darwinovej evolučnej teórie, používajúce priamu analógiu prirodzených genetických procesov. Predstavujú netradičný prístup k hľadaniu optimálneho alebo suboptimálneho riešenia zložitých optimalizačných problémov, ktoré nie sú riešiteľné klasickými technikami. Sú pravdepodobne najčastejšie používanými optimalizačnými algoritmami, s množstvom aplikácií od optimalizácie vysoko multimodálnych funkcií, až po riešenie kombinatorických a grafovo-teoretických problémov [23]. S úspechom sú používané najmä vtedy, keď hľadáme také globálne minimum, resp. maximum, ktoré je obklopené množstvom lokálnych miním, resp. maxím. Genetické algoritmy sú univerzálnou metódou pre simuláciu evolúcie, v ktorej je pre riešenie konkrétneho problému potrebné modifikovať len spôsob kódovania potenciálneho riešenia (štruktúru chromozómu) a spôsob výpočtu jeho sily.

Napriek tomu, že genetické algoritmy sú založené na paradigme prirodzenej evolúcie, je pri nich postup presne opačný ako v biológii. Najprv sa totiž definuje sila jedinca (jedinec je tým silnejší, čím je lepším riešením problému) a následne, na základe jej hodnoty, je jedincovi umožnená reprodukcia, prípadne je z populácie odstránený. Indivíduá s väčšou silou môžu vstupovať s väčšou pravdepodobnosťou do procesu *reprodukcie*, ktorá môže obsahovať ešte aj určitý *náhodný faktor* (tzv. genetický drift), napr. *náhodnú mutáciu* (v reťazci popisujúcom jedinca je náhodne vybraný symbol zamenený iným náhodne vybraným symbolom). Mutácia vykonáva malú zmenu genetickej informácie, aby sa zachovala rôznorodosť populácie a aby sa vniesla nová informácia. Táto skutočnosť umožňuje evolúcii hľadať nové riešenia, ktoré sa v populácii ešte vôbec nevyskytli a môžu byť nádejné pre ďalšiu evolúciu populácie. Celá nová populácia potenciálnych riešení je teda vytvorená výberom najlepších jedincov zo súčasnej generácie, ktorým je potom umožnená reprodukcia, aby vytvorili nových a prípadne lepších potomkov. Týmto spôsobom sa dobré vlastnosti indivíduí rozšíria po mnohých generáciách do celej populácie. Ak je teda genetický algoritmus dobre navrhnutý, celá populácia alebo aspoň jej časť bude konvergovať ku optimálnemu riešeniu daného problému.

B.3. Základné princípy genetických algoritmov

Genetický algoritmus v súčasnosti patrí medzi najpopulárnejšie evolučné optimalizačné algoritmy [15], [20]. Základné princípy genetických algoritmov sú určitou špecifikáciou všeobecných princípov evolúcie a jej algoritmizácie. Klasický genetický algoritmus je znázornený na obr. B.1.

Ak na riešenie daného problému chceme použiť genetický algoritmus, musíme najprv zvoliť vhodné *kódovanie* tohto problému, ďalej musíme definovať tzv. *fitness funkciu* (silu jedinca), ktorá každému potenciálnemu riešeniu priradí jeho kvalitu a napokon potrebujeme špecifikovať *operátory reprodukcie a mutácie*.

```
STANDARD-GA ()
1   vygeneruj počiatočnú populáciu
2   vypočítaj silu každého jedinca
3   while (not Finished) do begin                  ▷ vyprodukovanie novej generácie
4      for i ← 1 to PopulationSize/2 do begin      ▷ reprodukčný cyklus
5         vyber dvoch jedincov zo starej generácie ▷ s väčšou pravdepodobnosťou
6                                                  ▷ výberu silnejších jedincov
7         rekombináciou týchto jedincov vytvor dvoch potomkov
8         vypočítaj ich silu
9         vlož oboch potomkov do novej generácie
10     end
11     starú generáciu nahraď novou
12     if (populácia skonvergovala) then
13        Finished ← TRUE
14 end
```

Obr. B.1: Klasický genetický algoritmus

B.3.1. Kódovanie

Predpokladáme, že potenciálne riešenie problému možno reprezentovať ako množinu parametrov. Tieto parametre, tzv. *gény*, sú zlúčené do postupností, nazývaných *chromozómy*. Chromozóm je pojem používaný najmä v biológii pre postupnosť génov. Genetická informácia (*genotyp*) jedinca v prírode, rovnako ako v mnohých evolučných algoritmoch pozostáva z jedného alebo viacerých chromozómov. Každý gén reprezentuje nejakú vlastnosť alebo „časť“ vlastnosti jedinca. Hodnoty, ktoré môže nadobudnúť sa nazývajú *alely*[1]. Pod *fenotypom* sa chápu samotné prejavy genotypu, teda vlastnosti toho ktorého jedinca. Nie sú to už zakódované genetické informácie, ako pri genotype, ale ich reálne prejavy. Sila jedinca závisí na kvalite fenotypu. Možno ju vypočítať na základe genotypu pomocou *fitness funkcie*. Na vyššom stupni abstrakcie teda môžeme namiesto populácie jedincov hovoriť o populácii chromozómov.

V genetických algoritmoch sa zvyčajne chromozómy reprezentujú binárnymi reťazcami pevnej dĺžky. Každá pozícia v reťazci prislúcha jednému génu a alela na každej pozícii môže byť buď 0 alebo 1. Pre niektoré problémy však binárna reprezentácia nie je postačujúca. To je jedným z hlavných dôvodov, prečo sa objavili nové typy reprezentácií, napr. reprezentácia pomocou reálnych čísel. Takéto „chromozómy“ nemajú v podstate nič spoločné s „klasickými chromozómami“ z genetiky, ale z matematického hľadiska je účel týchto reprezentácií taký istý – majú obsahovať informáciu, ktorá charakterizuje jedinca (riešenie), a preto netreba zavádzať žiaden nový pojem.

[1] Veľmi jednoduchým príkladom môže byť napríklad to, že určitý gén reprezentuje „farbu očí“ a alela pre tento gén je napríklad „modré oči“, teda jedinec má „modré oči“.

B.3.2. Fitness funkcia

Pre každý riešený problém je nutné definovať tzv. fitness funkciu, ktorá konkrétnemu chromozómu (genotypu) priradí jednu numerickú hodnotu (zvyčajne kladné reálne číslo), ktorá istým spôsobom popisuje silu alebo kvalitu jedinca charakterizovaného týmto chromozómom. Mala by byť definovaná tak, aby najlepší chromozóm (t.j. chromozóm s maximálnou hodnotou fitness funkcie) bol riešením problému, ktoré hľadáme. Aby sa zvýšila pravdepodobnosť toho, že genetický algoritmus nám poskytne riešenie vysokej kvality, je vhodné, aby jedince, ktoré sú bližšie optimu, mali väčšiu silu ako tie, ktoré sú od neho ďalej. Prakticky však skoro nikdy nemáme potrebné informácie na to, aby sme mohli skonštruovať fitness funkciu spĺňajúcu túto podmienku. Pre mnoho problémov, konkrétne pri optimalizácii nejakej funkcie, sa ako fitness funkcia často používa samotná optimalizovaná funkcia, ktorá je prípadne trochu upravená. Pri väčšine problémov, napríklad kombinatorických, však bohužiaľ tento prístup nemožno zvoliť.

B.3.3. Operátory

Pre každú reprezentáciu (spôsob kódovania potenciálnych riešení) musia byť nad priestorom chromozómov[2] definované nasledujúce operátory: *selekcia*, *rekombinácia*, *mutácia* a *náhrada*. Počas fázy reprodukcie musia byť najprv vybrané[3] jedince (selekcia), na ktoré bude v rámci procesu reprodukcie aplikovaný najprv rekombinačný operátor a následne operátor mutácie. Pre chromozómy v tvare binárnych reťazcov sa tieto operátory zvyčajne definujú podobným spôsobom ako v prírode. Pri takto definovanej rekombinácii dochádza k priamej výmene genetickej informácie medzi dvoma chromozómami. V jednoduchšej podobe ide o tzv. *jednobodové kríženie* kedy sa pre dané dva chromozómy náhodne určí v ich reťazci deliaci bod, za ktorým sa všetky alely medzi týmito chromozómami vymenia. Keby sme sa však spoliehali len na kríženie, tak už po niekoľkých stovkách generácii by došlo k vystriedaniu všetkých možných kombinácií chromozómov a algoritmus by uviazol v lokálnom extréme a nevedel by sa pohnúť ďalej. Je to spôsobené tým, že kríženie nevnáša do genotypu žiadny nový materiál, ale len využíva to, čo už evolúcia vymyslela. Preto je potrebné použiť ďalší nemenej dôležitý operátor, operátor *mutácie*. Ide o jednoduchú operáciu nad chromozómovým reťazcom, pri ktorej sa s určitou pravdepodobnosťou náhodne vygeneruje bod, v ktorom dôjde k invertovaniu bitu (v prípade, že ide o reprezentáciu pomocou binárnych reťazcov). Aj tu existujú rôzne variácie mutačného operátora.

Príroda je významnou inšpiráciou, ale to neznamená, že musíme prevziať každý z týchto operátorov z reálneho sveta. Použitím operátorov, ktoré sú odlišné od tých, čo možno odpozorovať z prírody, môžno učiniť veľa z teórie genetických algoritmov zrozumiteľnejším a jednoduchším.

[2] Priestor jedincov sa zvykne stotožňovať s priestorom chromozómov, takže pojmy jedinec a chromozóm budeme často bez upozornenia zamieňať.

[3] Mali by byť uprednostňované najmä silnejšie jedince, ale vstup do reprodukcie by mal byť umožnený aj slabším jedincom, hoci s menšou pravdepodobnosťou.

B.3.4. Kritéria ukončenia

Optimálna funkčná hodnota riešeného problému je zvyčajne neznáma a je veľmi ťažké identifikovať stav, v ktorom by bolo najvhodnejšie ukončiť beh genetického algoritmu. Pre problémy, v ktorých je hodnota optima známa alebo hľadané riešenie spĺňa určité vopred známe podmienky, algoritmus možno zastaviť vtedy, keď ich dosiahne. Pre iné problémy treba nechať algoritmus pracovať nejaký konečný čas za súčasného kontrolovania, či celá populácia nestratila rôznorodosť. Nižšia rôznorodosť populácie totiž znamená menšiu pravdepodobnosť toho, že získame lepšie riešenie ako doteraz nájdené. Kritéria ukončenia berúce do úvahy pokles rôznorodosti populácie možno definovať mnohými spôsobmi. Jedným zo spôsobov je tzv. ε-konvergencia, pri ktorej algoritmus zastaví ak frekvencia nejakej alely na pozícii prislúchajúcej každému z génov je väčšia ako $1 - \varepsilon$, kde ε je kladná konštanta, zvyčajne veľmi malá, takže algoritmus zastaví, keď pre každý gén sú skoro všetky alely rovnaké.

Zoznam použitých symbolov a skratiek

Symboly

$\mathbb{N}$	množina prirodzených čísel
$\mathbb{N}_0$	množina prirodzených čísel vrátane nuly
$\mathbb{Z}$	množina celých čísel
$\mathbb{Z}^2$	kartézsky súčin $\mathbb{Z} \times \mathbb{Z}$
$\mathbb{C}$	množina komplexných čísel
$\mathbb{R}$	množina reálnych čísel
$\mathbb{R}^+$	množina kladných reálnych čísel
$[a_1, \ldots, a_n]$	usporiadaná n-tica prvkov $a_1, \ldots, a_n$
$f[\,\cdot\,]$	to isté ako $f([\,\cdot\,])$, ak f je funkčný symbol
$\mathbf{x}_k$	k-ty prvok postupnosti, resp. vektora $\mathbf{x}$
$\mathbf{x}_{k_1,k_2}$	prvok matice, prípadne dvojrozmernej postupnosti na pozícii k_1, k_2
a^*	číslo komplexne združené ku číslu a
$\mathbf{x}^*$	vektor, ktorého zložky sú komplexne združené ku zodpovedajúcim zložkám vektora $\mathbf{x}$
$\mathrm{Re}\{x\}$	reálna časť x
$\mathrm{Im}\{x\}$	imaginárna časť x
$O(f(n))$	trieda funkcií asymptoticky menších alebo rovných $f(n)$
$\Theta(f(n))$	trieda funkcií asymptoticky rovných $f(n)$
$\Omega(f(n))$	trieda funkcií asymptoticky väčších alebo rovných $f(n)$
$o(f(n))$	trieda funkcií asymptoticky menších ako $f(n)$
$\omega(f(n))$	trieda funkcií asymptoticky väčších ako $f(n)$
$\lg(x)$	logaritmus čísla x pri základe 2
$\mathrm{ri}(x)$	$\mathrm{Re}\{x\} + \mathrm{Im}\{x\}$
$\mathrm{cas}(x)$	$\cos(x) + \sin(x)$
C_j	$\cos(2\pi k_j n_j / N_j)$
S_j	$\sin(2\pi k_j n_j / N_j)$
$\aleph_0$	mohutnosť množiny prirodzených čísel (nulté kardinálne číslo)
$\prec, \succ$	rozšírené porovnanie čísel
$\mathbf{x} \cdot \mathbf{y}$	súčin vektorov, resp. matíc po zložkách
P	trieda problémov riešiteľných deterministicky v polynomiálnom čase

NP	trieda problémov riešiteľných nedeterministicky v polynomiálnom čase
$\mathbf{\Phi}(\mathbf{x})$	jednorozmerná diskrétna Fourierova transformácia vektora **x** (str. 6)
$\mathbf{\Phi}^{-1}(\mathbf{x})$	inverzná jednorozmerná diskrétna Fourierova transformácia vektora **x** (str. 6)
$\mathbf{\Phi}_{2\mathrm{D}}(\mathbf{x})$	dvojrozmerná diskrétna Fourierova transformácia matice **x** (str. 7)
$\mathbf{\Phi}_{2\mathrm{D}}^{-1}(\mathbf{x})$	inverzná dvojrozmerná diskrétna Fourierova transformácia matice **x** (str. 7)
$\overline{h}(\mathbf{x})$	jednorozmerná diskrétna Hartleyho transformácia vektora **x** (str. 7)
$\overline{h}^{-1}(\mathbf{x})$	inverzná jednorozmerná diskrétna Hartleyho transformácia vektora **x** (str. 8)
$\overline{h}_{2\mathrm{D}}(\mathbf{x})$	diskrétna cas-cas transformácia matice **x** (str. 11)
$\overline{h}_{2\mathrm{D}}^{-1}(\mathbf{x})$	inverzná diskrétna cas-cas transformácia matice **x** (str. 12)
$\mathbf{x} * \mathbf{y}$	diskrétna konvolúcia postupností **x** a **y** (str. 13)
$\mathbf{x} \otimes \mathbf{y}$	(diskrétna) cyklická konvolúca vektorov (matíc) **x** a **y** (str. 14)
μ	operátor zrkadlenia (str. 20)
$M \oplus N$	Minkowského súčet množín M a N (str. 23)
$\overline{M}$	doplnok množiny M (str. 23)
$M + \mathbf{t}$	posunutie množiny o vektor **t** (str. 23)
$-M$	symetrický obraz množiny M (str. 23)
$\mathrm{O}(M,N)$	množina takých posunutí **t**, že $M + \mathbf{t}$ a N sa prekrývajú (str. 28)
$\mathrm{D}(M,N)$	množina takých posunutí **t**, že $M + \mathbf{t}$ a N sa stýkajú (str. 28)
$\Upsilon(x)$	jednotková Heavisidova funkcia, jednotkový skok (str. 32)

Skratky

1D	jednorozmerný
2D	dvojrozmerný
DFT	(jednorozmerná) diskrétna Fourierova transformácia
DHYT	(jednorozmerná) diskrétna Hartleyho transformácia
FHYT	rýchla Hartleyho transformácia
2D-DFT	dvojrozmerná diskrétna Fourierova transformácia
DCCT	diskrétna cas-cas transformácia
FCCT	rýchla cas-cas transformácia
GA	genetický algoritmus

Literatúra a internetové odkazy

[1] ARNDT, J.: *Remarks on FFT Algorithms.*, september 1997. 34 s. K dispozícii na `http://www.jjj.de/fxt/`.

[2] BAILEY, D. H.: *FFTs in External or Hierarchical Memory.* NASA RNR Technical Report RNR-89-004, April 1989, 15 s.

[3] BEASLEY, D. — BULL, D. R. — MARTIN, R. R.: *An Overview of Genetic Algorithms: Part 1, Fundamentals.* University Computing, *15*, 1993, č. 2, s. 58–69.

[4] BEASLEY, D. — BULL, D. R. — MARTIN, R. R.: *An Overview of Genetic Algorithms: Part 2, Research Topics.* University Computing, *15*, 1993, č. 4, s. 170–181.

[5] BOUNSAYTHIP, C. — MAOUCHE, S.: *A Genetic Approach to a Nesting Problem in Textile Manufacturing Industry.* Proceedings of the 2NWGA, Vaasa, August 1996, 16 s. K dispozícii na `ftp://ftp.uwasa.fi/cs/2NWGA/Bounsaythip.ps.Z`.

[6] BOŽEK, M.: *Rozmiestňovanie geometrických útvarov*. Poznámky z prednášok 1998.

[7] BRIGHAM, E. ORAN: *The Fast Fourier Transform*. New Jersey, Prentice-Hall 1974. 252 s.

[8] DANIELS, K. — MILENKOVIC, V. J. — ROTH, D.: *Finding the maximum area axis-parallel rectangle in a polygon*. In: Proceedings of the 5th Canadian Conference on Computational Geometry. 1993, s. 322–327.

[9] DANIELS, K. — MILENKOVIC, V. J.: *Column-Based Strip Packing using Ordered and Compliant Containment*.

[10] DANIELS, K. — MILENKOVIC, V. J.: *Multiple Translational Containment: Approximate and Exact Algorithms*. In: Proceedings of the 6th Annual ACM-SIAM Symposium on Discrete Algorithms. 1995, s. 205–214.

[11] DANIELS, K. — MILENKOVIC, V. J.: *Multiple Translational Containment, Part I: An Approximate Algorithm*. Algorithmica, special issue on Computational Geometry in Manufacturing. Jún 1994. 47 s.

[12] DAWKINS, R.: *Sobecký gen*. Praha, Mladá fronta 1998. 320 s. Preklad z angličtiny: The Selfish Gene, 2nd ed., Oxford University Press 1989.

[13] ENGINEERING PRODUCTIVITY TOOLS LTD.: *The FFT Demystified (version 2.1)*. K dispozícii na `http://www.eptools.com/`, október 1999.

[14] FRIGO, M. — JOHNSON, S. G.: *FFTW User's Manual (version 2.1.2)*. Massachusetts Institute of Technology, k dispozícii na `http://theory.lcs.mit.edu/~fftw`, 1999.

[15] GOLDBERG, D.: *Genetic Algorithms in Search, Optimization, and Machine Learning*. Addison-Wesley, Reading, MA, 1989.

[16] GONZALES, R. C., — WINTZ. P.: *Digital Image Processing*. 2nd ed. Addison-Wesley, Reading, MA, 1987. 503 s.

[17] HARTE, T. P. — HANKA, R.: *Number theoretic transforms in neural network image classification*. Cambridge, CB2 2SR, 1997. 6 s.

[18] *The Hitch-Hiker's Guide to Evolutionary Computation: A List of Frequently Asked Questions (FAQ)*, USENET: `comp.ai.genetic`, `ftp://rtfm.mit.edu/pub/usenet/news.answers/ai-faq/genetic/`, 1998. 115 s.

[19] HLAVÁČ, V. — ŠONKA, M.: *Počítačové vidění*. Praha, Grada 1992. 272 s.

[20] HOLLAND, J. H.: *Adaptation in Natural and Artificial Systems*. The University of Michigan Press, Ann Arbor, MI, 1975.

[21] JAROSLAVSKIJ, L. — BAJLA, I.: *Metódy a systémy číslicového spracovania obrazov*. Bratislava, Alfa 1989. 526 s.

[22] KUŠNIER, M.: *Interaktívne rozmiestňovanie geometrických útvarov v nepravidelnej rovinnej oblasti*. [Diplomová práca.] Bratislava 1998. – Univerzita Komenského. Matematicko-fyzikálna fakulta. 48 s.

[23] KVASNIČKA, V. A KOL.: *Evolučné algoritmy*. Bratislava, Malé Centrum, v tlači.

[24] KVASNIČKA, V. A KOL.: *Úvod do teórie neurónových sietí*. Bratislava, IRIS 1997. 285 s.

[25] LI, Z. — MILENKOVIC, V.: *Compaction and Separation Algorithms for Non-Convex Polygons and Their Applications*. European Journal of Operations Research, *84*, 1995, s. 539–561.

[26] LI, Z. — MILENKOVIC, V.: *The Complexity of the Compaction Problem*. In: Proceedings of the 5th Canadian Conference on Computational Geometry. Waterloo, Canada, 1993, s. 7–11.

[27] LIPSON, J. D.: *Elements of Algebra and Algebraic Computing*. Addison-Wesley, Redwood City, California, 1981. 342 s.

[28] MILENKOVIC, V. — DANIELS, K., — LI, Z.: *Placement and Compaction of Nonconvex Polygons for Clothing Manufacture*. Fourth Canadian Conference on Computational Geometry, St. John's Newfoundland, August 1992.

[29] MILENKOVIC, V. — DANIELS, K., — LI, Z.: *Automatic Marker Making*. In: Proceedings of the 3rd Canadian Conference on Computational Geometry, Simon Fraser University, Vancouver B. C., August 1991, s. 243–246.

[30] MILENKOVIC, V. J.: *Multiple Translational Containment, Part II: Exact Algorithms*. Algorithmica, special issue on Computational Geometry in Manufacturing. Jún 1994. 28 s.

[31] MOSCATO, P.: *On Evolution, Search, Optimization, Genetic Algorithms and Martial Arts: Towards Memetic Algorithms*, Caltech Concurrent Computation Program, C3P Report 826, 1989.

[32] PELIKÁN, M.: *Genetic Algorithms by Means of Marginal Distributions*. [Diplomová práca.] Bratislava 1998. – Univerzita Komenského. Matematicko-fyzikálna fakulta. 77 s.

[33] POLEC, J. A KOL.: *Číslicové spracovanie signálov II*. Bratislava, Faber 1997. 155 s.

[34] POLEC, J. A KOL.: *Metódy kompresie obrazu*. Bratislava, Združenie používateľov telekomunikácií Slovenska 1998. 145 s.

[35] PRESS, W. H. A KOL.: *Numerical Recipes in C*, The Art of Scientific Computing, 2nd ed., Cambridge University Press, 1992. 994 s.

[36] Стоян, Ю. Г. — Гиль, Н. И.: Методы и алгоритмы размещения плоских геометрических объектов. Киев, Наукова думка 1975. 247 s.

[37] WALL, M.: *GAlib: A C++ Library of Genetic Algorithm Components (version 2.4)*, Documentation Revision B. K dispozícii na `http://lancet.mit.edu/ga/`, august 1996.

[38] WHITLEY, D.: *A Genetic Algorithm Tutorial*, Colorado State University, Dept. of CS, TR CS--93-103. K dispozícii na `http://www.cs.colostate.edu`, október 1999. 37 s.