B. Bezstratová kompresia zvuku

Každý pozná komprimačné programy, ako Rar, Zip, resp. ich rôzne verzie a odvodeniny. Keď nimi niečo skomprimujeme, očakávame dve skutočnosti: výsledná veľkosť súboru (súborov) bude nižšia (zväčša omnoho nižšia) ako pôvodná veľkosť, navyše budeme schopní presne rekonštruovať pôvodné údaje vďaka dekompresii, ktorú tieto programy tiež ponúkajú. Prvá vlastnosť v skutočnosti nie je zaručená – a predsa v praxi vždy nastáva, keďže dáta, s ktorými sa vo svete počítačov stretávame, majú zväčša nejakú charakteristiku, teda nesprávajú sa ako "náhodný šum". Druhá vlastnosť je tá "bezstratovosť", ktorej sa teraz budeme venovať.

Prečo sme sa vôbec zaoberali návrhmi stratovej kompresie, keď existuje aj bezstratová? Ako sme si už povedali, stratová kompresia sa opiera o sluchové vnímanie človeka, ktoré nie je dokonalé a je odlišne citlivé na zvuky s rôznymi charakteristikami. Ale keď existuje bezstratová kompresia... Problém je, že zatiaľ čo pre stratovú kompresiu si aspoň približne môžeme zvoliť kompresný pomer (teda aj veľkosť výsledného súboru – samozrejme, čím nižšie ju zvolíme, tým menej sa bude výsledok podobať originálu, alebo inými slovami, čo je dôležitejšie, výsledná kvalita bude nižšia), v prípade bezstratovej kompresie nemáme na výber – v rámci daného algoritmu môžeme len občas nastaviť hĺbku hľadania podobností, ale s výslednou veľkosťou nemôžeme natoľko manipulovať ako v prípade stratovej kompresie zvuku. Zatiaľ čo pri stratovej kompresii možno použitím kvalitných algoritmov dosiahnuť zvuk takmer neodlíšiteľný od originálu aj pri kompresnom pomere 1:12, v prípade bezstratových kompresií je to typicky 1:2. Možno je namieste otázku zo začiatku tohto odseku uviesť v presne opačnom význame: Aký význam má vlastne bezstratová kompresia, keď tu máme kvalitnú a efektívnu stratovú kompresiu?

B.01. Opodstatnenie bezstratovej kompresie

Zatiaľ čo stratová kompresia je priam určená pre nenáročných koncových používateľov, do segmentu profesionálnej práce so zvukom, jeho archivácie a šírenia sa nehodí. Dôvodom je, samozrejme, stratovosť. V prípade spracovania zvuku (napr. v nahrávacích štúdiách) by pri viacnásobnom spracovaní zvuku stratovými kompresnými algoritmami – keďže sú tieto založené na zakrývaní psychoakusticky ťažšie vnímateľných zložiek zvuku – bol výstupný zvuk už natoľko odlišný od pôvodnej nahrávky, že by zrejme začali vystupovať do popredia rôzne zvukové interferencie ako rekonštrukčné chyby, pôvodne kompresným algoritmom potlačené. Podobne je na tom archivácia – keď už archivovať, tak originál, nie veľmi podobnú kópiu. Trochu lepšie je to v prípade šírenia zvuku (napr. rozhlas, internetové rádio), kde sa bezstratová kompresia ponúka skôr ako alternatíva.

Je ale taký nízky kompresný pomer (1:2) vôbec výhodný? Určite je to významné ušetrenie prostriedkov – úložnej kapacity, resp. vysielacieho pásma. Druhou výhodou je integrita dát, keďže bezstratové algoritmy pri dekompresii dát zistia prípadné odlišnosti od originálu vďaka kontrolnému súčtu, ktorý zväčša obsahujú.

Značnou nevýhodou však ostáva výpočtová náročnosť – algoritmy bezstratovej kompresie zvuku zvyknú byť veľmi náročné na procesorový čas (rádovo zložitejšie než stratová kompresia zvuku), čo ich znova posúva do roly špeciálneho použitia, ktoré nie je určené pre masy. Je však jasné, že táto prekážka je len dočasná – rýchly rast výpočtových schopností nielen počítačov, ale i špecializovaných stolových prehrávačov, dokonca malých prenosných prehrávačov či mobilov totiž sľubuje skorú možnosť implementácie v praxi.

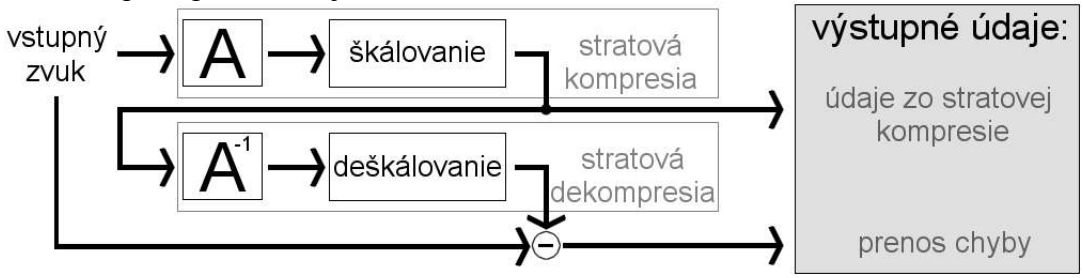
V nasledujúcej časti sa pozrieme na dva základné princípy, s ktorými sa pri návrhu algoritmu na bezstratovú kompresiu zvuku v praxi stretávame.

B.1. Bezstratová kompresia ako doplnok kompresie stratovej

Tento prístup sa môže čitateľovi zdať logický – máme algoritmy na stratovú kompresiu. Keďže sa tieto snažia skomprimovať vstup tak, aby bol veľmi podobný originálu, môžeme ich využiť na základnú kompresiu vstupných dát. Tento prístup sme si už opísali tu*. To, čo zostane, bude rozdiel medzi originálom a tým, čo by sme dostali po dekompresii dát zo stratovej kompresnej schémy – akási chybová zložka, zvyšok transmisie. Tento prístup rozvíja napr. Liebchen[b.1.01] a nazýva ho bezstratové transformačné kódovanie (*verlustlose Transformacionscodierung*). Faktom však ostáva, že v praxi sa používa menej.

B.1.01. Bezstratové transformačné kódovanie

Bezstratové transformačné kódovanie sa zakladá najmä na diskrétnej transformácii. Vstupný zvuk je najprv stratovo skomprimovaný, tak príslušným algoritmom znova dekomprimovaný. Rozdiel medzi pôvodným a týmto zvukom je potom "chybová zložka" - skreslenie, ktoré stratová kompresia spôsobila. Táto je prenášaná (resp. ukladaná) spolu s výsledkom samotnej stratovej kompresie. Tento postup znázorňuje schéma:



obr. b.1.01.1 – Príklad bezstratového transformačného kódovania – algoritmus stratovej kompresie hrá dôležitú rolu

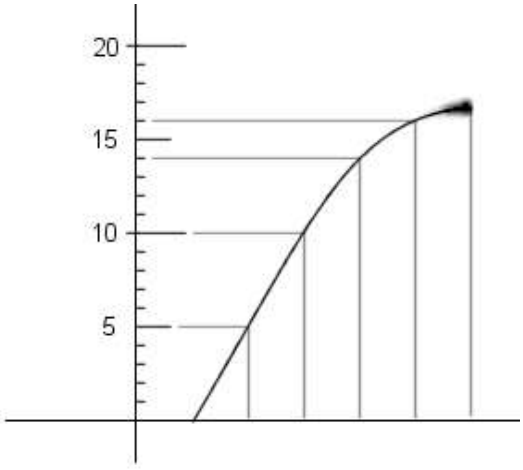
Pozornému čitateľovi určite neušiel tento problém: akú stratovú kompresiu je vhodné zvoliť? Ak bude použitá kompresia s vysokým dátovým tokom, výsledná chyba bude veľmi malá – ale veľkú časť výstupných údajov bude tvoriť práve výstup z algoritmu na stratovú kompresiu. Ak zvolíme nízky dátový tok, chyby budú vyššie, navyše ostanú korelované. Voľba škálovacieho faktora, ako sme si ho popísali v sekcii o stratovej kompresii zvuku, musí byť nanovo preskúmaná. Liebchen a kol.[b.1.02] v svojej práci najprv ukazujú, že z pohľadu celkovej entropie výstupu je najvhodnejšie voliť škálovacie faktory v rozmedzí $0 < \alpha << 1$ (v závislosti od charakteru komprimovaného zvuku), ktoré budú zhodné na celej šírke frekvenčného spektra, no pre prvú verziu algoritmu LTC (*linear transformation coding*) volia natvrdo $\alpha = 1$. Dôvodom je ľahké kódovanie chyby, ktorá nadobúda hodnoty 0,1,-1. Vo svojej diplomovej práci[b.1.01] neskôr Liebchen volí škálovací faktor $\alpha = 0{,}25$ (pričom tento prístup používa aj v algoritme LTAC[b.1.01]).

Úlohou teda ostáva:
- Zvoliť vhodnú transformáciu A a škálovanie pre algoritmus stratovej kompresie.
- Navrhnúť vhodné kódovanie pre výstup zo stratovej kompresie a vhodné kódovanie na prenos chyby. Tento problém rozoberieme v kapitole B.4.

Rôzne algoritmy majú rozličný prístup k týmto problémom, preto sa teda ich riešením budeme zaoberať pri ich rozbore.

B.2. Bezstratová kompresia založená na predikcii

Tento prístup využíva vlnový charakter zvuku. Zvuk totiž nie je zmesou náhodného šumu, ale má isté zákonitosti, "hladkosť". (Ak by sme sa pohybovali v spojitom priestore, hovorili by sme o spojitosti zvuku v čase. Keďže je však zvuk, s ktorým digitálne pracujeme, diskrétny – aj keď vznikol kvantizáciou spojitých údajov – zvuk bude vždy trochu "zúbkovitý".) Ak je napríklad hodnota za sebou idúcich vzoriek 0, 5, 10, 14, 16, dá sa očakávať, že nasledujúca hodnota bude blízko čísla 17. Príklad vidno na obrázku:



obr. b.2.01 - Zvuk je spojitý, preto možno hodnotu nasledujúcej vzorky predpovedať

Kódovanie zvuku pomocou tohto prístupu k nemu – teda keď sa na základe niekoľkých predošlých hodnôt vzoriek snažíme pomocou nejakej lineárnej funkcie "uhádnuť" hodnotu nasledujúcej vzorky – sa nazýva *kódovanie lineárnou predikciou* (*LPC*). Tento prístup je využívanejší (aj viac preskúmaný) ako transformačné kódovanie. Navyše, aj MPEG-4 ALS (*Advanced Lossless System*)* je na založený práve na ňom.

B.2.01. Lineárna predikcia

Tento prístup je založený na skutočnosti, že hodnotu nasledujúcej vzorky možno odhadnúť funkciou, ktorej vstupnými parametrami je p posledných hodnôt vzoriek. Takáto predikcia má potom rád p. V praxi sa používajú lineárne funkcie, keďže sú jednoduché a dosahujú uspokojivé výsledky. Vysvetlíme si použitie predikcie na príklade spomenutom vyššie:

Označme $X(i)$ hodnotu vzorky v čase i. Nech $P(i)$ je hodnota predikovaná (predpovedaná) pre vzorku v čase i. Majme nasledujúce hodnoty vzoriek ($X(0)$ .. $X(4)$): 0, 5, 10, 14, 16. Máme viacero možností, ako predpovedať $X(5)$. Použime napr. prediktor druhého rádu tvaru $P(i)=2X(i-1)-X(i-2)$. Pre $i=5$ je to teda $P(5)=2X(4)-X(3)=2.16-14=18$. Tento prediktor druhého rádu sa zjavne snaží zachovať rozdiel medzi susednými hodnotami, keďže $P(5)-X(4)=X(4)-X(3)=2$.

Prediktory sú zväčša volené natvrdo – to, čo môžeme ušetriť zvolením „vhodnejších" prediktorov by sme zväčša minuli na prenos jednotlivých koeficientov. Preto napr. v technickej dokumentácii k programu Shorten[c.1.03] vidíme nasledujúce prediktory:

$P_0(i)=0$
$P_1(i)=X(i-1)$
$P_2(i)=2X(i-1)-X(i-2)$
$P_3(i)=3X(i-1)-3X(i-2)+X(i-3)$

Keby sme použili napr. $P_3(i)$ v spomenutom príklade, dostali by sme: $P_3(i)=3X(4)-3X(3)-X(2)=316-3.14+10=16$ – číslo, ktoré je s najväčšou pravdepodobnosťou bližšie k hádanému číslu ako predošlé číslo 18.

Natíska sa otázka: čo sme tým získali? Dekoreláciu zdroja. Miesto ďalšej hodnoty budeme jednoducho prenášať len rozdiel ďalšej hodnoty a hodnoty predikovanej. Túto "chybu" budeme nazývať ***reziduál*** (alebo tiež chybový, resp. reziduálny signál, ak hovoríme o časovom priebehu). Efektívnym kódovaním reziduálov (napr. cez Riceove kódy) získame menší dátový tok, než aký mal zdroj, keďže chyby budú štatisticky veľmi malé, s hustým rozdelením okolo nuly.

B.2.02. Všeobecné riešenie lineárnej predikcie

Pre všeobecné koeficienty a_0, a_1, ..., a_p pre predikciu rádu p chceme dosiahnuť, aby v ideálnom prípade pre čím viac vzoriek $X(i)$ platilo:

$\sum_{k=1}^{p} a_k X(i-k)=X(i)$, resp. snažíme sa voliť taký rád a koeficienty, aby pre celý blok bola štandardná odchýlka skutočných hodnôt vzorky od hodnôt predikovaných čím nižšia. Takáto sústava (pre rôzne i a p) rovníc, resp. sústava rovníc s odchýlkou sa rieši pomocou Levinson-Durbinovho algoritmu[b.2.01], teda je otázkou hrubej výpočtovej sily (beží v čase $O(p^2)$). Je zrejmé, že na prvých p vzoriek nemôžeme použiť tento prístup – preto môže byť jedným z prístupov[b.2.02] tzv. progresívna predikcia, t.j. postupné zvyšovanie rádu predikcie, kým tento nedosiahne p zvolené pre daný blok.

B.2.03. Autokorelácia

http://en.wikipedia.org/wiki/Linear_prediction
to do – FIR kódovanie http://en.wikipedia.org/wiki/FIR_filter
finite impulse response, ale aj infinite: http://en.wikipedia.org/wiki/Infinite_impulse_response
GSM, speech coding – to do

B.2.04. Kódovanie reziduálov

Laplaceovo rozloženie, Huffmanove kódy, Riceove kódy, rozsahové kódovanie – to do
http://svr-www.eng.cam.ac.uk/reports/ajr/TR156/node6.html
Golombovo kódovanie: http://www.ecs.csun.edu/~dsalomon/DC2advertis/p53.pdf

B.2.05. Adaptívna predikcia

Adaptívnosť predikcie sa môže prejaviť na dvoch úrovniach:

1. *Adapcia rádu predikcie*. Pre zvolený blok (ktorý môže byť rôznej veľkosti, čo je tiež jedným z prejavov adaptívnosti, ako sme si už povedali) sa vykoná kompresia pre rôzne rády predikcie. Nakoniec sa použije ten, ktorý prinesie najväčšiu úsporu.
2. *Adapcia predikovateľnosti*. Keďže vstup nie je dokonale predikovateľný (to by sme potom nepotrebovali ani prenos chyby), môžeme meniť "faktor spoliehania sa na predikciu". Na začiatku by mal hodnotu napr. $m=0{,}75$. To znamená, že ako ďalšie $P(i)$ sa použije hodnota $P(i-1)+m.(P_c(i)-P(i-1))$, pričom $P_c(i)$ je hodnota, ktorú vráti algoritmus predikcie. V našom príklade by sme miesto čísla 18 získali číslo ale $16+(18-16).0{,}75=17{,}5$. Chyba, za predpokladu, že by nasledovalo číslo 17, by teda bola -0,5. V ďalšom kroku by teda algoritmus predikcie upravil m smerom bližšie k nule,

keďže nižší prediktor by bol dal menšiu chybu. Úpravou faktora predikcie možno dosahovať lepšie výsledky, než bez neho. (Pozn: Čísla v tejto kapitole sú len ilustračné, prediktor zväčša pracuje s celými číslami.)

B.3. Výber algoritmu pre bezstratovú kompresiu

Na fóre *hydrogenaudio*[b.3.01] bola v auguste 2004 urobená anketa s otázkou: "ktorý bezstratový zvukový kodek preferujete?" Výsledky boli nasledovné (zotriedené podľa popularity):

Kodek:	počet hlasov:	podiel:	podiel rok predtým[b.3.02]:
FLAC:	254	53,36%	54,9%
Monkey's Audio:	95	19,96%	30,5%
WavPack:	52	10,92%	5,5%
Apple lossless:	24	5,04%	N/A
WMA Lossless:	8	1,68%	3,0%
OptimFROG:	7	1,47%	1,2%
TTA:	7	1,47%	N/A
La:	4	0,84%	3,0%
Shorten:	0	0,00%	1,2%
iné, resp. nezáujem o bezstratovú kompresiu:	25	5,25%	N/A

(Treba spomenúť, že tento prieskum určite nie je reprezentatívny, keďže prebehol v rámci špecifickej komunity ľudí.) Používatelia mali rozličné kritériá – pre niekoho bola najdôležitejšia výsledná veľkosť súboru, pre niekoho rýchlosť kódovania, pre niekoho rýchlosť prehrávania. Pre mnohých to bola podpora vo viacerých operačných systémoch, dokonca platformách (Apple), resp. podpora v hardvérových prehrávačoch. Veľkým plus bolo zahrnutie projektu do *Open Source*, resp. "životnosť" kodeku – keď už príliš dlho nebol nijak vylepšovaný a opravovaný, ľudia naň zabúdali.

[b.1.01] LTAC (Lossless Transform Audio Compression) je kodek vyvinutý Tilmanom Liebchenom z Technickej univerzity v Berlíne ako súčasť diplomovej jeho práce (http://www.nue.tu-berlin.de/wer/liebchen/docs/Diplom.pdf) v rokoch 1997-1998. V praxi sa nedočkal veľkého úspechu, preto ho nebudeme ďalej rozoberať. Ďalšie zdroje možno nájsť na http://www.nue.tu-berlin.de/wer/liebchen/ltac.html, a [b.1.02]

[b.1.02] M. Purat, T. Liebchen, P. Noll: Lossless Transform Coding of Audio Signals. 102nd AES Convention, Munich, 1997 (http://www.nue.tu-berlin.de/wer/liebchen/docs/aes102.pdf).

[b.2.01] Pozri napr. http://noel.feld.cvut.cz/~pollak/m/links/levdurb.m. a iné*

[b.2.02] Pozri článok Extended Linear Prediction Tools for Lossless Audio Coding (T. Moriya, D. Yang, T. Liebchen), publikovaný na IEEE ICASSP 2004, Montreal, May 2004 (dostupný na http://www.nue.tu-berlin.de/Publikationen/papers/0301008_ICASSP2004_Moriya_etal.pdf).

[b.3.01] http://www.hydrogenaudio.org/forums/index.php?showtopic=24921.

[b.3.02] http://www.hydrogenaudio.org/forums/index.php?showtopic=12050.