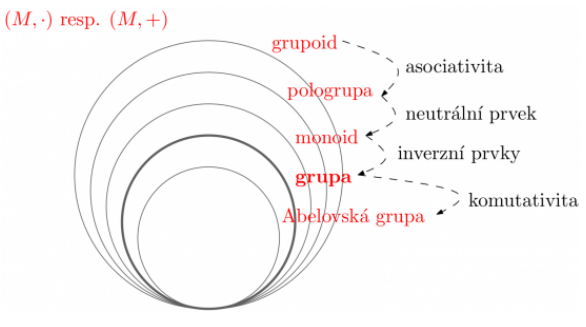
MI-SZZ

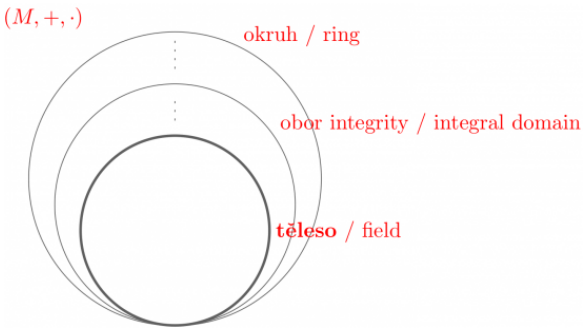
Jakub Waller, Tomáš Malíček

12. června 2017

Obsah



Obrázek 1: Hierarchie množin s binární operací a jejich „ANIK“ vlastnosti.



Obrázek 2: Hierarchie okruhů, oborů integrity a těles. T–OB-OK = „TOBOK“.

Část I
Společné okruhy

1 MPI

1.1 Teorie grup: Grupoidy, pologrupy, monoidy a grupy. Podgrupy, cyklické grupy a jejich generátory.

- Grupoid: $(M,*)$, kde M je množina uzavřená vůči operaci $*$
- Pologrupa: asociativní grupoid
- Monoid: pologrupa s neutrálním prvkem
- Grupa: monoid s inverzníma prvkama
- Abelovská grupa: komutativní grupa
- Podgrupa: podmnožina množiny tý grupy, která spolu se stejnou operací tvoří grupu
  - Triviální podgrupy: prázdná, a samotná grupa. Zbytek jsou netriviální (vlastní) podgrupy.
  - Neutrální prvek a inverzní prvky jsou ty samý jako v původní grupě
- Řád grupy = počet prvků
- Cyklické grupy
  - $\langle N \rangle$ – podgrupa generovaná množinou N.
  - Grupa $G = (M,*)$ je cyklická, pokud existuje prvek $a \in M$ tak, že $\langle a \rangle = G$. Prvek a je generátor grupy G.
  - Každá cyklická grupa je taky abelovská
  - Pokud a je gener. cykl. grupy řádu n, pak a^k je generátor multiplikativní grupy pokud $gcd(k,n) = 1$
  - Každá cyklická grupa má $\varphi(n)$ generátorů – eulerovská funkce = počet menších nesoudělných čísel
  - Podgrupy cyklické grupy jsou cyklické
- Lagrangeova věta: Jestliže je H podgrupa G, dělí řád podgrupy H řád (konečné) grupy G. Důsledkem je, že grupa prvočíselného řádu má pouze triviální podgrupy.
- Malá Fermatova věta: V cyklické grupě $G = (M,*)$ řádu n platí pro všechny prvky $a \in M$, že $a^n = e$, kde e je neutrální prvek.
- Homomorfismus = Takový zobrazení z grupoidu do grupoidu, že je jedno, jestli nejdřív provedu operaci z G a pak výsledek zobrazím do grupoidu H, nebo jestli nejdřív zobrazím prvky z G do H a pak na nich provedu operaci z H. Izomorfismus - prostý a na.

1.2 Tělesa a okruhy: Základní definice a vlastnosti. Konečná tělesa. Okruhy polynomů, ireducibilní polynom.

- $(M, +, *)$ je okruh, pokud $(M, +)$ je Abelovská grupa, $(M, *)$ je pologrupa a platí distributivní zákon
- $a * b = 0$ se nazývají dělitelé nuly. Komutativní okruh bez dělitelů nuly je obor integrity.
- Okruh $(M, +, *)$ je těleso, jestliže $(M \setminus \{0\}, *)$ je grupa.
- Příklad okruhu, co není těleso: $(\mathbb{Z}, +, \cdot)$
- Konečně těleso = těleso, které má konečný počet prvků. Základní příklad je množina zbytkových tříd modulo prvočíslo p, $\mathbb{Z}_p = \{0, 1, \ldots, p-1\}$.
  - Řád $(\mathbb{Z}, +)$ je p a každý nenulový prvek je generátor.
  - Řád $(\mathbb{Z}, *)$ je $p-1$, je cyklická a má $\varphi(p-1)$ generátorů.
  - Řádem konečného tělesa musí být p^n, kde p je prvočíslo a n přirozené číslo. p se nazývá charakteristika. Všechna tělesa řádu p^n jsou navzájem izomorfní.
- Komutativní okruh polynomů nad okruhem K (sčítání a násobení polynomů snad zvládneme), značen $K[x]$.
- Buď $P(x) \in K[x]$ stupně alespoň 1. Řekneme, že $P(x)$ je ireducibilní nad K (aka prvočíslo mezi polynomy), jestliže pro každé dva polynomy $A(x)$ a $B(x)$ z $K[x]$ platí $A(x).B(x) = P(x) \Rightarrow (stupen A(x) = 0 \vee stupen B(x) = 0)$.
  - Příklad polynomu ireducibilního nad reálnými čísly: $x^2 + 1$
- Binární těleso je těleso s 2^n prvky (koeficienty jsou modulo 2) a značí se $GF(2^n)$
  - Sčítání po složkách modulo 2
  - Násobení polynomů modulo nějaký zvolený ireducibilní polynom stupně n
  - Použití: AES
- Konstrukce těles řádu p^n:
  - $n = 1$. Množina $\mathbb{Z}_p$ s operacemi sčítání a násobení modulo p.
  - $n > 1$. Množina polynomů z okruhu $\mathbb{Z}_p[x]$ stupně nejvýše $n-1$ s binárními operacemi sčítání po složkách modulo p, násobení v okruhu $\mathbb{Z}_p[x]$ modulo zadaný polynom stupně n ireducibilní nad $\mathbb{Z}_p$

1.3 Funkce více proměnných: gradient, Hessián, definitnost matic. Extrémy funkcí více proměnných a jejich hledání. Hledání vázaných extrémů (pouze s rovnostními vazbami).

- Funkce f má v bodě $x \in R$ derivaci, je-li f definovaná v okolí bodu x a existuje-li limita $f'(x) = \lim_{h \to 0} \frac{f(x+h)-f(x)}{h}$
- Parciální derivaci funkce $f(x_+, ..., x_n)$ vzhledem k x_i v bodě $a = (a_1, ..., a_n) \in D_f$ můžeme definovat jako: $\frac{\partial f}{\partial x_i}(a_1, ..., a_n) = \lim_{h \to 0} \frac{f(a_1,...,a_i+h,...,a_n)-f(a_1,...,a_n)}{h}$
- Gradient - směr nejrychlejšího růstu funkce f, $\bigtriangledown f = (\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}, \frac{\partial f}{\partial z})$
  - Pro všechny lok. extrémy platí, že gradient se rovná nule, jsou tzv. stacionární body.
- Jacobiho matice - matice parc. derivací vektorové funkce f, řádek určuje funkci a sloupec určuje které číslo z vektoru x se použije pro parc. derivaci, tato matice je zobecnění gradientu a pro jednu funkci je rovna gradientu.
- Hessova matice - matice parc. derivací druhých řádů, řádek určuje první proměnnou podle které se derivuje a sloupec tu druhou, determinant této matice se jmenuje Hessián (někdy se tak označuje sama tato matice).
- Definitnost matic
  - Pozitivně (pro $\forall \vec{x} \in R^n \setminus \{\vec{0}\}$ platí $\vec{x}^T M \vec{x} > 0$)
  - Negativně (pro $\forall \vec{x} \in R^n \setminus \{\vec{0}\}$ platí $\vec{x}^T M \vec{x} < 0$)

  - Poz. semidefinitní (pro $\forall \vec{x} \in R^n$ platí $\vec{x}^T M \vec{x} \geq 0$ a ex. nenulové $\vec{y} \in R^n$ splňující $\vec{y}^T M \vec{y} = 0$)
  - Neg. semidefinitní (pro $\forall \vec{x} \in R^n$ platí $\vec{x}^T M \vec{x} \leq 0$ a ex. nenulové $\vec{y} \in R^n$ splňující $\vec{y}^T M \vec{y} = 0$)
  - Indefinitní (zbytek, tzn. právě když existují vektory $\vec{x}, \vec{y} \in R^n$ splňující $\vec{x}^T M \vec{x} > 0$ a $\vec{y}^T M \vec{y} < 0$)
  - Sylvestrovo kritérium - slouží k jednoduššímu určení zda je matice poz. či neg. definitní, platí jen pro symetr. matice
    * Poz. definitní, když determinanty všech podmatic začinajících vlevo nahoře jsou kladné
    * Neg. definitní, když determinanty všech podmatic začínajících vlevo nahoře jsou nenulové, střídají znaménko, a první je záporný

- Extrémy funkcí více proměnných a jejich hledání - zkoumáme body kde je gradient roven nule, tam může být min., max. nebo sedlový bod, který to je se zjistí určením definitnosti Hessovy matice v těchto bodech
  - Jestliže matice poz. def., pak je v bodě ostré lok. ostré min., pokud semidef., pak min. není ostré.
  - Jestliže matice neg. def., pak je v bodě ostré lok. ostré max., pokud semidef., pak max. není ostré.
  - Jestliže matice indef., pak je v bodě sedlový bod
  - Glob. extrémy se hledají: z různých počátků, stochasticky, restartováním algoritmu s jinou poč. konfig. vzhledem k již nalezeným bodům.
- Hledání vázaných extrémů s rovnostními vazbami - máme nějakou rovnici jako podmínku
  - Hledáme lok. extrémy Lagrangeovy funkce $\Phi(x, y, \lambda) = f(x, y) + \lambda g(x, y)$ (pro 2 proměnné), kde g je ta podmínková rovnice
  - Postup: najdeme stacionární body pomocí gradientu, vyřešíme soustavy rovnic a tím dostanem stac. bod(y), ty dosadíme do Hessovy matice a klasicky pomocí determinantů této matice zjistíme jestli lok. max. nebo lok. min.
- Ptal se na ten grafický význam gradientu, chtěl vědět, kde že jako je, to sem taky moc nevěděl, chtěl slyšet, že je to vektor v tom definičním oboru té fce.

1.4 Integrál funkce více proměnných.

- Motivace: výpočet objemu/obsahu pod grafem funkce, objem těles, hledání těžíště...
- Riemannův určitý integrál - aproxiamce plochy pod křivkou pomocí obdélníků
  - interval rozdělíme na malé kousky (tzv. rozdělení intervalu), na těchto kouscích aproximujeme funkci $f(x)$ vhodně zvolenými konstantami, dostaneme stupňovite funkce, obsah pod grafem stupň. funkce je součet obdélníků a tedy snadno spočítatelná veličina, přesnou hodnotu získáme tak, že v limitě pošleme šířku uvedených obdélníků k nule
  - Vlastnosti: aditivita integrálu, multiplikativita integrálu
- Newtonova-Leibnizova formule - definice určitého integrálu založená na existenci primitivní funkce, její hledání je víceméně inverzní proces k derivování, $\int_a^b f(x)dx = [F(x)]_a^b = F(b) - F(a)$
- Dvojný integrál nad obdélníkovou oblastí - převedeme na dva 1D problémy, $\int_a^b (\int_c^d f(x, y)dy)dx$, nejdříve zintegrujeme vzhledem k jedné proměnné a druhou považujeme za konstantu, poté provedeme druhou integraci a to už je tam jen jedna proměnná
- Dvojný integrál nad obecnou oblastí - místo c a d se dosadí nějké funkce a dělá se to samé jako nad obdélníkovou oblastí, vysvětlení je takové, že víceméně nasčítáváme úzké obdélníky (co jsou na výšku mezi těmi funkcemi), stejně se to dá dělat pokud je omezené spojitými funkcemi x a ne y
- Trojný integrál a aplikace - analogické konstrukci integrálu dvojného, pouze integrujeme funkci tří proměnných, $\int\int\int f(x, y, z)dxdydz$, výpočet lze opět převést na 3 výpočty 1D integrálu, existuje ovšem 3! možných pořadí integrování.

2 PAR

2.1 PRAM modely a algoritmy.

- RAM model - výpoč. jednotka, vst. a výst. páska, neomez. # paměť. buněk lokální paměti
  - Instrukce pro přesun dat, aritm., logické a větvící trvající jednotkový (konst.) čas
  - Čas. slož. algoritmu = počet provedených instrukcí, Prostorová slož. alg. = počet použitých paměť. buněk
- PRAM model - neomezený počet procesorů RAM (bez pásek, ale s vlastní lokální pamětí)
  - neom. # sdílených pam. buněk přístupných v jedn. čase, vstup a výstup a komunikace procesorů přes sdílenou paměť
  - Operace: čtení sdíl. buňky (R), lokální operace (L), zápis do buňky sdíl. paměti (W).
  - proc. #1 má aktivační registr active/idle pro každý proc., výpočet trvá dokud se #1 nezastaví (ten se zastaví pokud ostatní skočily)
  - Paralelní časová složitost: jednotkový model (vše trvá 1), globální model (L trvá 1, W a R trvá $d > 1$).
  - Význam PRAM modelu: silný (přístup k jakék. buňce sd. paměti v konst čase), jednoduchý a intuit., jako zkušební model.
  - Ošetření konfliktů při přístupu do sd. paměti:
    - EREW PRAM - žádné dva proc. nemohou R ani W téže buňky sd. paměti najednou
    - CREW PRAM - může se současně R, ale ne W
    - CRCW PRAM - oboje se může současně, typy: prioritní (proc. mají priority, zápis je povolen proc. s nejv. prioritou), náhodný (vybere se náh. proc.), shodný (všem žádajícím proc. je povoleno dokončit zápis pokud jsou zapisované hodnoty stejné).
  - Výpočetní síla - A je výp. silnější než B ($A >= B$), když jakýk. alg. napsaný pro B poběží beze změny na A s tímtéž paral. časem, při stejných architektonických parametrech
  - Pro CRCW platí: priority>=arbitrary>=common>=CREW>=EREW.
  - PRAM alg. je časově efektivní pokud $O(log^{O(1)}(n))$ a zároveň $C(n,p) = O(SU(n) * log^{O(1)}(n))$.
- PRAM algoritmy:
  - Paralelní hledání - pro prezentaci různých přístupů do sdílené paměti
  - PRAM algoritmy s konst. časem/plně paralelní:
    - Shodný CRCW PRAM $(n+1, n)$ alg. pro výpočet $OR(x_1, ..., x_n)$, $p = n$
    - Shodný CRCW PRAM $(2n+2, n^2)$ alg. pro výpočet nejlevějšího maxima $max(x_1, ..., x_n)$, $p = n^2$ procesorů mřížce indexovaných s konst. časem
    - Shodný CRCW PRAM $(n + \sqrt{n} + 1, n)$ alg. pro výpočet indexu prvního výskytu hodnoty 1 v binárním poli X, p=n.

2.2 Přímé ortogonální a řídké hyperkubické a nepřímé vícestupňové propojovací sítě pro paralelní počítače.

- Pojmy: stupeň uzlu, k-regulární graf, kartézský součin grafů, uzlo a hranově symetrický graf, excentricita uzlu, průměr grafu, poloměr grafu, souvislost grafu, bipartitní graf, Hamilt. kružnice, bisekční šířka
- Přímé propojovací sítě ortogonální
  - Binární hyperkrychle Q_n - 2^n uzlů, $n2^{n-1}$ hran, průměr n, bisek. šířka 2^{n-1}
    - regulární (vš. uzly mají stejný stupeň a tedy stejný počet uzlů), log. stupeň uzlů, hierarchicky rekurzvní, uzlová a hran. symetrie, největší možná bisek. šířka, vzd. uzlů daná počtem rozdílných bitů
  - Mřížka $M(z_1, ..., z_n)$ - $\prod_{i=1}^{n} z_i$ uzlů, $\sum_{i=1}^{n}((z_i - 1)\prod_{j=1,j!=i}^{n} z_j)$ hran, průměr $\sum_{i=1}^{n}(z_i - 1)$
    - 2D a 3D nejpraktičtější pro použití, není regulární (není uzlově sym.), velký průměr, heirarchicky rekurzivní, bipartitní, Hamilt. cesta vždy ex., částečně ale neefek. škálovatelná
  - Toroid $K(z_1, ..., z_n)$ - mřížka, kde je každá lin. řada uzavřena do kružnice

    - regulární, uzlově sym., bipart. pokud jsou všechny délky stran sudé, částečně škálovatelný (ještě méně efektivní v porovnání s mřížkami)

- Přímé propojovací sítě řídké hyberkubické - řídké grafy odvoz. od hyperkrychle
  - Zabalený motýlek wBF_n - $n2^n$ uzlů, $n2^{n+1}$ hran, bis. šířka 2^n
  - Obyč. motýlek oBF_n - $(n+1)2^n$ uzlů, $n2^{n+1}$ hran, průměr $2n$, bis. šířka 2^n
    - není regulární ani uzlově symetrický, heirarchicky rekurzivní

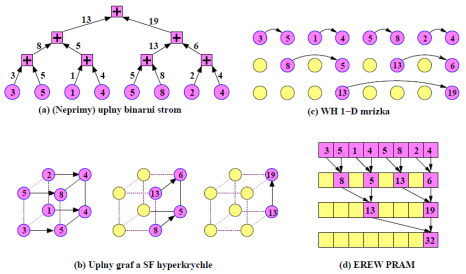
- Nepřímé vícestupňové sítě (MIN): Banyan NxN MIN, k-ární delta MIN, obecná k-ární MIN
  - Druhy permutačních stupňů:
    - Dokonalé promíchání (Perfect shuffle) σ - rotace vlevo o 1 bit
    - Motýlek (Butterfly) β_i - záměna posledního a i-tého bitu
    - Základní (Baseline) δ_i - rotace doprava posledních $i+1$ bitů, bity před tím nechávám beze změny

2.3 Paralelní redukce a prefixový výpočet nad polem, paralelní konstrukce Eulerovy cesty v grafu.

- Třída NC ("Nick's Class") je množina jazyků rozhodnutelných v nejvýše polylogaritmickém čase $T(n, p(n)) = O(\log^{O(1)} n)$ na počítači PRAM s nejvýše polynomiálním počtem procesorů $p(n) = O(n^{O(1)})$.

- Paralelní redukce = provedení binární operace se všemi prvky pole (např je všechny sečtu)
  - Je optimální na hyperkubických sítích
  - WH lineární pole, hyperkrychle, EREW PRAM: jednoduše přímo se složitostí $O(\log n)$
  - mřížky: po dimenzích

Implementace paralelní redukce



  -

- Prefixový součet = pro pole X je výstupem pole Y, kde Y[i] je součet prvních i prvků pole X. stejná škálovatelnost jako redukce
  - EREW PRAM – Pro predstavu, bunka „pricte“ svoji hodnotu do bunky o 1 vedle, v dalsim kromu o 2, pak o 4, ... ve stylu 2^{krok}
  - Nepřímý strom výšky h(T) – počítá 2h(T) kroků, vyšle to nahoru a pak dolu
  - Přímý strom výšky h(T) – na něj namapováno pole A n vstupních hodnot v pořadí POSTORDER, opět 2h(T) kroků. Stejné jako nepřímý, jen ještě nelisty k tomu přičtou svoji hodnotu.
  - Hyperkrychle – vysílání všichni všem (zelený registr pro součet všech čísel, žlutý registr pro prefixový součet)
  - Mřížky a toroidy – hodnoty se pošlou do pravého sloupce, pak v tomto sloupci do všech řádků a nakonec dojde k doinformování v řádcích (směrem doleva)

- Segmentovaný prefixový součet
  - vstupní pole rozděleno do segmentů, úkolem je spočítat prefixový součet v rámci jednotlivých segmentů
  - pokud má pravej prvek před sebou zábranu (tj. ty nalevo od něj patří do jinýho segmentu), tak nahoru do vyššího uzlu leze jen ten prvek (místo součtu levýho a pravýho)

- Konstrukce Eulerovy cesty v grafu
  - Eulerovská cesta vede přes všechny hrany právě jednou

- Pole uzlů Adj → pole hran AA (hrana, následník a anti-paralelní dvojče)
- Pro každou hranu e v AA (do in parallel): ET[e] := AA[e].sib.next
- slouží jako základ pro široké spektrum paralelních výpočtů, např. výpočet minimální kostry, testování planarity, testování souvislosti grafu, ...

2.4 Paralelní řadící sítě, 0-1 lemma, mřížkové a hyperkubické paralelní algoritmy pro řazení.

- základní operací je porovnej-a-vyměň (C&E), sekvenční algoritmus má složitost O(n · log n)
- Nepřímá síť:
  - složená ze sloupců komparátorů (HW implementací operace C&E)
  - počet paralelních C&E kroků = hloubka sítě
  - je statická, realizuje datově necitlivý algoritmus
  - pokud N > k, na každý vstup frčí $\frac{N}{k}$ čísel a komparátory provádějí Merge-and-Split
- Přímá síť:
  - C&E operace probíhají mezi dvojicí procesorů
  - Topologie: mřížky, toroidy, hyperkrychle, hyperkubické sítě, ...
  - Přímý řadící alg. = posloupnost dokonalých párování procesorů odvozených z jejich lineárního očíslování (výběr očíslování má vliv na složitost). Např v mřížce: po řádcích, po sloupcích, hadovitě
  - Také datově necitlivé
- 0-1 řadící lemma: Jestliže datově necitlivý řadící algoritmus dokáže setřídit libovolnou binární vstupní posloupnost, pak dokáže setřídit libovolnou vstupní posloupnost. Důkaz nejlépe viz 8. přednáška, slajdy 8 a 9., nevím, jestli musíme znát.
- Řazení na mřížkách:
  - 1D – paralelní bubblesort: sudo-liché C&E na lineárním poli
  - 2D – ShearSort: logn + 1 řádkových hadovitých fází a logn sloupcových fází
  - 3D – 3DSort: komplikované aka čekuj fitwiki nebo přednášky
- Řazení na hyperkubických sítích:
  - Základem je MergeSort – sudolichý, sudosudý, bitonický (Batcherovy algoritmy). Realizace: řadící nepřímá síť, motýlek, hyperkrychle, simulace na mřížkách

3 SPI

3.1 Odhady parametrů statistických rozdělení a modelů, odhady hustoty a distribuční funkce.

- Centrální limitní věta = suma několika nezávislých a stejně rozdělených (iid) náhodných veličin se přibližuje k normálnímu rozdělení s přibývajícím počtem veličin (náh. veličiny jsou např. kostky, čím víc jich je, tím víc součet toho co jsem hodil zapadá do gausovky)
- Konfidenční interval (interval spolehlivosti pro střední hodnotu) - má nějakou přesnost, tozn. když vybereme samply z populace, tak by jejich střední hodnota měla být s určitou pravděpodobností v tomto intervalu
  - Známe rozptyl σ^2 nebo je velikost výběru n větší než 30: normální rozdělení, $\overline{X}_n \pm q_{1-\frac{\alpha}{2}} \frac{\sigma}{\sqrt{n}}$
  - Neznáme rozptyl: studentovo rozdělení s výběrovým rozptylem s_n^2, $\overline{X}_n \pm \tau_{\frac{\alpha}{2};n-1} \frac{s_n}{\sqrt{n}}$
  - Jednostranný/oboustranný odhad
- Odhad hodnot parametrů pravděpodobnostních rozdělení:

  - Momentová metoda: Obecný moment $\mu_i = m_i'$ výběrovému momentu (ten se získá, spočítá). Metoda založena na zákonu velkých čísel = pro hodně pokusů se experimentální charakteristiky blíží teoretickým. 1. moment = střední hodnota μ, 2. moment = rozptyl σ^2
  - Věrohodnostní (likelihood) funkce = sdružená hustota pravděpodobnosti $f(x_1|\theta) \cdot f(x_2|\theta) \cdot \ldots$ - pravděpodobnost, že naměřená data pasují do rozdělení s parametry θ.
  - Metoda maximální věrohodnosti = maximalizujeme funkci nahoře (aka derivace) – nebo její logaritmus, to je egál. Chci největší pravděpodobnost, že naměřená data pasují do daných modelů.

- Histogram = odhad pravděpodobnostní funkce (graf) - either četnosti nebo relativní četnosti
- Kernelový odhad hustoty = pomocí parametru h vyhlazuje křivku funkce hustoty.
- Empirická distribuční funkce = je schodovitá, se zvětšující se velikostí náhodného výběru n se blíží teoretické distribuční funkci. Sečtou se veličiny v náhodném výběru menší než x a vydělí velikostí náh. výběru.
- Gaussian mixtures: $f_x(x) = p_1 f_1(x) + p_2 f_2(x)$. Příklad: pravděpodobnosti p_1, p_2 určujou, jestli je to muž/žena, distribuce f_1, f_2 pak říkají, jak vysoký(á) je.
- EM (expectation-maximisation) algorithm = mám samply z několika distribucí a chci zjistit, který kam patří a jaký distribuce to jsou. Přiřadím nějak ty samply do distribucí, a mrknu, jak tam pasujou (estimate fáze – spočítám loglikelihood funkci). Poupravím to přiřazení (maximise fáze – maximalizuju tu loglikelihood funkci) a zase mrknu. Atd atd až to konverguje. Funguje takhle třeba k-means.

3.2 Statistické testy hypotéz o parametrech modelů, t-testy, testy nezávislosti, testy dobré shody.

- Nulová vs alternativní hypotéza, buď můžeme zamítnout nebo nezamítnout nulovou hyp.
- Chyba prvního (zamítnuta a platí) vs druhého druhu (nezamítnuta a neplatí)
- Studentovo t-rozdělení umožňuje dělat přijatelné závěry i na základě málo dat: jednovýběrový, dvouvýběrový (na dvou různých skupinách) a párový (dvojice měření na stejné skupině) t-test, který testuje pomocí střední hodnoty to jak jsou si podobná rozdělení.
  - Jednovýběrový: $T = \frac{\overline{X} - \mu_1}{s_X}\sqrt{n}$. H_0 zamítáme pokud $|T| > t_{1-\frac{\alpha}{2};n-1}$. Zvolíme střední hodnotu μ_1 základního souboru (kterou chceme zamítnout, abychom řekli že tam nepatří ten výběr), ověřujeme hypotézu zda pokusný výběrový soubor pochází z populace.
  - Párový: $T = \frac{\overline{Z} - d}{s_Z}\sqrt{n}$, kde $\overline{Z} = \overline{X} - \overline{Y}$ a $d = \mu_1 - \mu_2, s_Z = s_X - s_Y$. Zvolíme μ_1 a μ_2.
- Test nezávislosti = koukáme, jestli náhodné proměnné jsou nezávislé.
  - Kanadský a googlitelný testy: runs test (kolik nad a pod), difference sign test (počet stoupajících úseček), turning point test (počet změn směrů)
  - Českej test: Runs above/below the mean – $R_i = 1$ pokud se hodnota v i-tém prvku mění z "nad μ"na "pod μ"nebo obráceně. $N_n = \sum_i R_i$. To by mělo odpovídat normálnímu rozdělení $N_n \approx N(\mu = \frac{n+1}{2}, \sigma^2 = \frac{n-1}{4})$. Vypočteme $Z_n = \frac{N_n - E(N_n)}{\sigma = \sqrt{varN_n}}$. Pokud Z_n zapadá do standardního normálního rozdělení, proměnné jsou nezávislé.
- Test dobré shody (Pearsonův chí-kvadrát test) = používá se v případech kdy neznáme rozdělení. Porovnává naměřenou frekvenci s teoretickou. Např. u kostky se dá zkoumat, jestli je cinklá. Viz hezkej příklad na wiki

3.3 Markovské řetězce s diskrétním a spojitým časem. Jejich limitní vlastnosti.

- Náhodný proces - soustava reálných náh. veličin indexovaná nejč. časem
- Poissonův proces - náh. proces se spoj. časem, který je čítací, homogenní (= má stacionární přírůstky) a bez paměti (má nezávislé přírůstky), má intenzitu λ
- Markovův řetězec - náh. proces, který splňuje Markovovu podmínku (je bez paměti).
- Markov. řet. s diskr. rozdělením (DTMC), definován: množ. stavů, vekt. poč. rozdělení, maticí přechodů

- dělí se na homog. (pravděpodobnosti přechodů se nemění) a nehomog., je ireducibilní pokud se dá dostat z lib. stavu do jiného (nemusí být v jednom kroku)
- Matice přechodů P - řádek odkud, sloupec kam, součet řádků je 1, P^n má vícekrokové pravd.
- Druhy stavů (p = pravd. že se znova tam vrátí): tranzientní ($p < 1$), rekurentní ($p = 1$), absorpční (nedá se opustit).
- Absopční řetězce - obsahuje mimo stavy tranzientní i absorpční
- Matice přechodů absorpčního Mark. řetězce ($tranz.stavymaticeQ, abs.stavymaticeR; nul.matice, jednot.matice$)
- Fundamentální matice absorpčního řet. $N = (jednotMatice - Q)^{-1}$ - kolikrát se proces v průměru ocitne v tranz. stavech, součet řádků je střední doba absorpce při daném počátečním stavu
- Pravd. matice že spadneme do stavu j když začínáme v i, $B = N * R$

- Markov. řet. se spojitým parametrem (CTMC, spojení DTMC s Pois. proc.) - 3 matice: přechodů P_t (součty řádků 1, obsahuje funkce kde je parametr t), skokových intenzit Q (součty řádků 0), U diskrétních přechodových pravděpodobností v náh. čase t (pro $i \neq j$ platí $\frac{\lambda_{i,j}}{\lambda_{MAX}}$, pro $i = j$ platí $1 + \frac{\lambda_{i,j}}{\lambda_{MAX}}$, λ_{MAX} je největší součet odchozích skok. intenzit, touto maticí převedeme CTMC na DTMC)
  - $Q = P'_t(t = 0)$, $P'_t = Q * P_t$, $P'_t = P_t * Q$, řet. musí být homogenní aby platily
  - Intenzita přechodu (Jump rate) - je dána součinem λ a přechodové pravděpodobnosti
  - Kolmogorov-Chapmanova rovnice - říka, že pravd. toho, že se z nějakého stavu dostaneme do jiného zjistíme tak, že sečteme pravd. průchodu přes všechny mezilehlé uzly

- Stacionární rozdělení (rovnovážná distribuce) - když se nechá běžet proces donekonečna, tak se ustálí v nějakém vektoru pravděpodobností.
  - Pro DTMC - vektor π se nazývá stacionární rozdělení, $\pi = \pi * P$ (splňuje podmínku detailní rovnováhy $\pi_i p_{i,j} = \pi_j p_{j,i} \forall i, j$ pak je stac. rozdělení), všechny prvky vektoru musí být nenulové a $\sum_{i=1} \pi_i = 1$
  - Pro CTMC - $\pi = \pi * P_t$ pro každé $t >= 0$ (také se dá použít podmínka detailní rovnováhy pro určení zda je to stac. rozd., stejná jako u DTMC), dále platí, že π je stacionární rozdělení právě tehdy, když $\pi * Q = 0$
  - Detailní rovnováha slouží k ověření, zda nějaké π je stacion. distr., rovnice (ty co jsou uvedené v předch. dvou bodech) může být příliš složitá, a proto existuje detailní rovnováha - tou můžeme ověřit nějaký odhad.

3.4 Systémy hromadné obsluhy, jejich limitní vlastnosti a stabilita. Souvislost s Poissonovým procesem a Markovskými řetězci.

- Obslužný systém obsahuje: vstupní tok požadavků, frontu, obsluhu, výstupní tok požadavků
- Kendallova notace: A (střední počet jednotek co vstoupí během čas. jednotky), B (střední počet obsloužených během čas. jednotky), X (# poč. paralleních serverů/kanálů/linek), Y (kapacita fronty, výchozí nekonečno), Z (chování/režim fronty, výchozí First-Come-First-Served).
- Zabýváme se těmito typy: $M/M/1$, $M/M/m$ (tento je postaven na Markovském B&D řetězci), $M/G/\infty$.
- Obslužný systém může být otevřený (prvky se po obsloužení nevracejí do systému) a uzavřený (vracejí se).
- A příchody - tvoří stochastický proces, rozdělení intervalů mezi příchody: Poiss. proc., pravidelné deter. příchody, Erlangovo rozložení intervalů mezi příchody, nespecifik. rozdělení, obecné & nezávislé.
- B výstupní tok - typy doby obsluhy: exponenciální rozložení, konstantní, Erlangovo rozložení, obecné/jakékoliv
- Z řežim fronty - typy: First-in-first-out, Last-come-first-served, First-in-last-out, Priority based, Random
  - Velikost fronty: nulová, neomez., omezená
  - Disciplína fronty: absolutně netrpělivá (do fronty se vůbec nezařadí pokud nemůže být hned obzloužen $->$ systém se ztrátami), bez netrpělivosti (prvky čekají nekonečně dlouho klidne $->$ systém s čekáním), částečně netrpělivá (prvek čeká po určitou dobu jen)
- Birth-and-death řetězce - spec. případ CTMC (má matici skok. intenzit Q), birth (přišel nový požadavek, systém přijímá s intenzitou λ), death (požadavek byl obsloužen, systém odbavuje s intenzitou μ)

  - Stacionární rozdělení π $B\&D$ Mark. řetězce je $\pi(n) = \frac{\lambda_{n-1}}{\mu_n}\pi(n-1)$ pokud platí $\sum_{n=0}^{\infty}\pi(n) = 1$ a $\forall n >= 1 : \pi(n) >= 0$, dále zde pro π platí i podmínky detailní rovnováhy.

- Charakteristiky systémů hromadné obsluhy $(M/M/m)$ - na jejich základě lze systém analyzovat
  - Míra vytížení $\rho = \frac{\lambda}{\mu}$ musí být $\rho < 1$ aby systém pracoval
  - Střední doba transakce (obsluhy) $T_S = \frac{1}{\mu}$
  - Střední počet zákazníků ve frontě $N_Q = \frac{\rho^2}{1-\rho}$
  - Střední doba čekání ve frontě $EW = \frac{\rho}{\mu-\lambda}$

- Littleova věta: Průměrný počet pož. ve stabilním systému je roven průměrné frekvenci příchodů vynásoben průměrným časem požadavků v systému - $N = \lambda T$, kde N je prům. počet zákaz. v systému, T je prům. (střední) doba, jakou zákazník stráví v systému.

4 PAA

4.1 Význam tříd NP a NPH pro praktické výpočty.

- Pojmy: kombinatorický problém, konfigurační proměnné, instance problému, konfigurace, řešení instance, certifikát, SAT, stav algoritmu, stavový prostor, strategie pohybu stav. prostorem (úplná, systematická)
- Typy komb. problémů: rozhodovací (zda existuje), konstruktivní (sestroj něco), enumerační (sestroj všechna), optimalizační verze těchto problémů, kdy se vyžaduje, aby nějaký atribut byl co nejlepší, navíc ještě optimalizační evaluační problém (např. zjisti nej. možnou cenu C odpovídající urč. konfiguraci)
- Deter. Turingův stroj - násl. stav lze jedn. určit na základě čteného symbolu a akt. stavu.
- Nedet. Turingův stroj - umožňuje v každém kroku rozvětvit výpočet na n větví.
- $P \subseteq NP$; $NPH \cap NP = NPC$; P versus NP problém
- Třída P - obs. vš. problémy řešitelné pomocí det. Tur. stroje v polyn. čase, tedy $O(n^k)$, kde n je vel. instance a k je konečné číslo., např. nalezení kostry grafu
- Třída NP - to samé akorát nedeter., případně výsledek se dá ověřit v polyn., např. nalezení Hamilt. kružnice
- Třída co-NP - inverzní k NP, nemáme certifikát, např. zda je graf prost Hamilt. kružnic
- Třída PO - např. problém nejkratší cesty v grafu
- Třída NPO - velikost výstupu instance je omez. polynomem ve vel. instance, např. optimal. verze TSP
- Problém x je NPH tehdy, pokud existuje NPC problém y takový, že je možné y redukovat na x v polyn. čase. U problému v NPH, který není v NPC, podle mě není možné ověřit řešení v polynomiálním čase.
- NPC jsou vzájemně převoditelné nejtěžší problémy v NP (SAT, 3-SAT, batoh).
- Třída NPI=NP-P-NPC - problémy, kde neumíme nalézt polyn. alg., ani na ně převést SAT
- Turingova redukce - deter. Tur. stroj, volání podprogramu
- Karpova redukce - deter. Tur. stroj, převedení na jiný problém
- Cookova věta: existuje NPC problém
- Jak dokážu, že je algoritmus NPC: najdu NP problém, kterej na tenhle NPC problém můžu převést (Karpova redukce)
- Nakonec zřejmě chtěl slyšet, že na NPH se hůře hledají heuristiky než na NP

4.2 Experimentální vyhodnocení algoritmů, zejména randomizovaných.

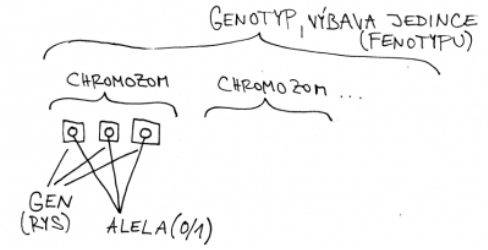
- Pseudopol. alg. - závisí polynomiálně na vel. instance, dále na paramteru, který s vel. instance nesouvisí.
- Aproximativní alg. - každou instanci vyřeší v polyn. čase v rel. chybou ϵ.
- SAT - součin součtů, zda je pro nějké ohodnocení pravdivé (MAX-SAT - kolik klauzulí lze splnit).
- Při neúnosném NPO problému máme na výběr: deter. metoda (zaručuje chybu v nejh. případě, patří: aprox. alg., pseudopol. alg.), náhodná a kombinovaná metoda (dává statistickou chybu v prům. případě, patří sem randomiz. alg.).
- Randomiz. alg. - nedeter. alg, rychlejší řešení těžko řešitelných problémů (např. NPC), různé výsledky, více spuštění, kromě klasického vstupu má ještě vstup random. čísel., např. MAX k SAT (Monte Carlo = MC), Miller-Rabinův test prvočís. (MC), Quicksort (LV)
  - 2 druhy rand. alg.: typu Monte Carlo (pevný čas. optimaliz. kritérium náhodné - může být ohraničeno), typu Las Vegas (přesné řešení, čas běhu náhodný - může být ohraničen).
  - Zabýváme se buď střední hodnotou kvality nebo času běhu alg. (v analýze/experimentech)
  - Výhody: strukt. jednoduchost, očekáv. kvalita výsledku může být lepší než zaruč. kvalita deter. pomalých alg., nezávislým opakováním se dá zlepšit kvalita
  - Experiment - plán exp. $->$ provedení exp. $->$ interpretace výsledků $->$ odpověď $->$ otázka $->$ plán exp.
    - Otázky: Je alg. A lepší než alg. B? Pro určité instance jestli je to pravda atp.
    - Hodnotíme: kvalitu řešení (kde známe řeš. absolutně, při heuristikách relativně), výp. nár. (záleží na stroji, nejlepší měřit počet testovaných stavů).
    - Jaké případy: nejhorší, nejlepší, průměrny, v závislosti na parametru.
    - Výběr instancí: náhodně gen. (škál. rovn. rozd.), náhodně gen. s ohledem na exp., standardní bench.
    - Typy hodnocení: white box (známý alg., limit. počet inst., detailní měř.), black box (nezn. alg., kompletní počet inst., měř statist. dat, ověření kvality, výkonu).
    - Práce s parametry: nutno respektovat závislosti, popř. otestovat závislosti, nastavení paramterů je te cesta dalším stav. prostorem.
    - Vyhodnocení: v záv. na vel. inst. (šum - nutno více měření), kritické zhodnocení (jsou hodnoty spolehlivé a vysvětlitelné), výstup (kvantitavní data, poté nalézáme zákonitosti - histogramy, bodové grafy, takto získáme kvalitativní informace).
    - Příklady: WalkSAT, 3-SAT, náhodná procházka.

4.3 Princip lokálních heuristik, pojem globálního a lokálního minima, obrana před uváznutím v lokálním minimu.

- Důležité pojmy: stav algoritmu, okolí stavu, k-okolí stavu, stavový prostor, sousední stav, graf stavového prostoru, heuristická funkce, greedy heuristika
- Lokální minimum = všechny sousední stavy mají horší hodnotu optimalizačního kritéria.
- Globální minimum = všechny stavy mají horší hodnotu optimalizačního kritéria.
- Heuristiky se dělí na globální a lokální (mají lokální minimum na párku).
- Od heuristiky očekáváme: použitelnost v praxi, možnost omezení a optimalizace (čas, přesnost), hledání přibližného nebo přesného řešení
- Druhy: konstruktivní, iterativní, dvojfázová
- Překonání lokálního minima: jednoduché (má to na párku), s návratem (z minima vycouvá), pokročilé
- Praktické řešení: zvětšení okolí stavu, start z několika různých počáteních řešení, vracení se (best-first, backtracking – nepolynomiální algoritmy), dočasné zhoršení (simulované ochlazování), více stavů najednou (genetické alg.), restrikce (tabu search)
- Příklady: náhodná procházka, best first (best only?), first improvement
- Dynamické programování pro batoh – dekompozice podle celkové ceny:

- $W(0,0) = 0$
- $W(0,c) = \infty$ pro všechna $c > 0$
- $W(i+1,c) = \min(W(i,c), W(i, c - c_{i+1}) + w_{i+1})$ pro všechna $i > 0$.
- Sloupce jsou indexy věcí, řádky jsou celková cena batohu, uvnitř tabulky = váha batohu

4.4 Princip genetických algoritmů, význam selekčního tlaku pro jejich funkci.

- Pojmy: Jedinec (fenotyp), genetická reprezentace (genotyp/chromozom), gen, alela, generace (populace), mutace, křížení, zdatnost (fitness), degenerace, konvergence, biodiverzita
- 

- Princip: inicializace, selekce, křížení, mutace, reprodukce
- Selekční tlak = pravděpodobnost výběru nejlepšího jedince
  - velký tlak - nebezpečí degenerace populace a uváznutí v lokálním minimu
  - malý tlak - pomalá konvergence
  - liší se podle výběrového mechanismu

5 TES

5.1 Signály, systémy a jejich vlastnosti, automat jako popis systému.

- Systém má obvykle: komponenty, sadu znaků, nekončící vývoj, diskrétní kroky
  - Abeceda = libovolná množina **S**, prvky = znaky
  - Signál v diskrétním čase na **S** = nekonečná posloupnost prvků z **S**
  - $\sum_{\mathbf{S}}$ = množina signálů na **S**
- Popis systémů - black box/vnitřní popis
- Black box = popis chování zvenku, relace mezi vstupními a výstupními signály. Vlastnosti:
  - Receptivní – pro každý vstupní signál existuje nějaký výstupní signál
  - Kauzální – výstup závisi v každém okamžiku jen na minulosti
  - Deterministický – každý vstup má přesně jeden výstup – relace se poté stává funkcí
  - Bez paměti – výstup závisí v každém okamžiku pouze na aktuálním vstupu
- Vnitřní popis reaktivních systémů
  - Automat (reprezentuje systém) je pětici – stavy, počáteční stavy, vstupy, výstupy, přechodová relace
  - Automat je deterministický právě když má přesně jeden počáteční stav, a když pro každý vstup a stav existuje právě jeden další stav a výstup.
  - Extended state machines = automaty s proměnnými (Př. Semafor: chodec zmáčkne tlačítko, po 30 vteřinách přepíná na zelenou:)
  - Hierarchické automaty = automaty s nadstavy a podstavy, rekurzivní skrývání/modelování podrobností.

5.2 Kompozice systémů a automatů (diskrétních, spojitých), synchronní reaktivní modely.

Kompozice systémů a automatů

- Synchronizovaná paralelní kompozice systémů – systémy dělají vždy jeden krok spolu. Přechod dvojice (vstup prvního, vstup druhého) → (výstup prvního, výstup druhého) patří do synchronizované kompozice systémů, právě tehdy když (vstup prvního) → (výstup prvního) patří do prvního systému a zároveň (vstup druhého) → (výstup druhého) patří do druhého.
- Synchronizovaná paralelní kompozice automatů – Stavy nového automatu jsou vždy zkombinované stavy skládaných automatů, čili dvojice (stav prvního, stav druhého). Stejné platí pro vstupy a výstupy. Přechod mezi stavy nového automatu patří do přechodové relace, pokud přechod mezi jejich prvními složkami je součástí prvního automatu, a přechod mezi druhými složkami součástí druhého (stejné platí pro složky vstupů a výstupů).
- Kaskádní kompozice systémů – výstup prvního systémů = vstup druhého systému
- Kaskádní kompozice automatů
- Obecná kompozice – synchronní reaktivní modely: libovolné propojení vstupů a výstupů (smyčky, více vstupů a výstupů, řízení toku).
  - Každá komponenta sítě musí obsahovat přechody mezi vstupem do ní (což může být přechod z jiné komponenty, nebo vstup sítě) a výstupem z ní (což může být přechod do jiné komponenty, nebo výstup sítě).
  - Roztrhání smyček = delay
  - Table Lookup (vyhledávácí tabulka)
- Synchronní reaktivní modely jako automaty

Spojité modelování času

- Diskrétní čas – stav bývá výsledkem minulého stavu. Spojitý – „minulý stav“ postrádá smysl. Místo toho vektorové pole – přiřazuje hodnotám reálných čísel směr, kterým se budou vyvíjet. Obyčejné diferenciální rovnice: $\dot{x} = f(x)$, přičemž f je vektorové pole.
- Nedeterminismus – obvykle pochází z okolí systému, neznámých detailů, detailů, které jsme nemodelovali.

5.3 Typologie ověření správnosti systémů (testing, bounded model checking, unbounded model checking a jejich základní principy).

- **Testing** - vybíráme konečnou množinu T konečných cest (testovací příklady), pro každé $t \in T$ ověřujeme, jestlí z t můžeme usoudit $\not\models \Phi$, tj. na základě t víme, že existuje protipříklad (cesta π tak, že $\pi \not\models \Phi$)
- Simulace - generování konečných cest
- **Bounded model checking** - omezené ověřování modelů
  - Cílem ověřování je zjistit, zda náš přechodový systém splňuje specifikaci Φ (to je LTL formule)
    - přechodový systém ji splňuje pokud ji splňuje pro každou cestu (posloupnost stavů)
  - Problémy s ověřováním: je třeba to ověřit (zda splňuje speficik. LTL formule) pro každou cestu, cest může být nekon. mnoho a nekon. délky $->$ ověř. jen definovaný prefix cest (prvních i stavů cesty) $->$ omezená (bounded) délka cesty
  - Ověřování modelů - je náročné ověřit všechny cesty $->$ snaha najít protipříklad
    - Globally ok = pro cestu délky n ověřujeme nějakou interpretaci I(ok) (např. {1,2,3}), tedy v cestě délky n nesmíme vstoupit do jiného stavu než z té množiny, jinak G ok neplatí
- **Unbounded model checking** - neomezené ověřování modelů - pro všechny cesty i nekon. délky (využití indukce)
  - Invariant - množina obsahující všechny cesty délky k, které jsou podmnožinou I() (= množina stavů, na které platí daná stavová vlastnost), dalo by se říci něco jako podmnožina všech řešení z množiny I()

- Induktivní invariant - pokud stav x invariantu splňuje podmínky a stav x' vznikl přechodem z x do x', pak x' ji také musí splňovat
  - jinými slovy: když se ze stavu A můžeme dostat do stavu B, tak stav musí splňovat také danou podmínku, pakliže ji nesplňuje, tak se nejedná o induktivní invariant
  - Jejich výpočet: množina V musí obsahovat počáteční stavy a žádny krok nesmi opustit V

5.4 Boolovská splnitelnost: algoritmy a jejich využití v bounded model checking.

- Pointa je v reprezentaci BMC (Bounded model checking) pomocí booleovské formule
  - př. formule $(P \wedge Q) \vee (\neg Q \vee P)$ reprezentuje $\neg BMC(Gok, n)$, vstupem je tedy místo BMC bool. formule, výstupem je splňující evaluace, nebo "unsat", pokud není splnitelná $->$ máme tzv. SAT řešič
- SAT řešič - vstupem bool. formule v CNF, výstupem splňující evaluace nebo unsat
  - naivní algoritmus vyzkouší 2^v možností (všechny kombinace v proměnných)
  - Simplifikace - uvažujeme pouze CNF formu (literál=proměnná či její negace, klausule=disjunkce literálů - or)
    - Pokud je v klauzuli jen 1 literál $->$ musí být true (je to unit penetration).
    - Vyskytuje-li se proměnná v klauzulích jen poz./neg., můžeme ji přiřadit hodnotu tak, aby klauzule mohli být T = pure literal elimination.
  - CDCL (Conflict driven clausule learning)
    - náhodně se ohodnocují proměnné, lze si představit jako strom, kde v každém uzlu se jde buď doleva nebo doprava (T nebo F) pro danou úroveň reprezentující konkrétní proměnnou
    - pokud vybrané ohodn. nikam nevedou, vrátíme se o několik úrovní výše a zkusíme jiné ohodn.
  - Lokální vyhledávání - neúplná heuristika, náhodně vybírá ohodnocení
    - postupně se snažíme zvýšit počet splněných klauzulí
    - často rychle najde řešení, ale není úplná $->$ neprozkoumá všechny možnosti $->$ nelze s ní dokazovat nesplnitelnost
  - řešiče se rychle vyvíjejí, každoročně soutěž řešičů, dnes dokáží řešiče řešit obrovské SAT problémy
- Případně tam frknout trochu PAA? Řešení SAT pomocí genetických algoritmů, heuristiky

Část II
Oborové okruhy – Znalostní inženýrství

6 PDD

6.1 Metody pro hodnocení relevance příznaků, heuristické metody pro výběr nejlepší podmnožiny atributů (FS), metody redukce počtu instancí (numerosity reduction).

- **Feature ranking and selection**
  - hodně dostupných algoritmů, vybírají most informative variables (features)
  - Hlavní použití: redukce dimenzionality datasetu, najití optimálního subsetu příznaků, řazení příznaků na základě jejich významnosti.
  - FS metody se dají rozdělit na univariate (zvažují jeden příznak v jednu chvíli) a multivariate (zvažuje subsety příznaků), mezi multivariate metody patří (mohou být i v univariate, pokud se testuje jen jeden příznak):
    - Filter metoda - řadí příznaky nebo jejich subsety nezávisle na prediktoru (klasifikátoru).
      - vyber příznaky, pak klasifikuj, robustní proti přeučení, používá různé statistické testy
    - Wrapper metoda - používá klasifikátor k ohodnocení příznaků či subsetu, rozdělení na training/validation/test dataset, používá crossvalidation
      - vytvoř subsety příznaků, klasifikuj, vyber nej. subset, náchylný k přeučení
    - Embedded metoda - jako u wrapperu, the search is controlled by the algorithm constructing classifier, používá crossvalidation
      - eliminuj nepotřebné příznaky tak dlouho dokud bude stejný výkon, výsledky podobné wrapperům, méně výp. náročné, méně náchylné na přeučení
  - Univariate metody - jak relevantní je proměnná x_i k predikci outputu - relevance, testy, závislost, korelace, pearsonův koeficient, mutual information, filtry, wrappery, embedded
  - FS metody (jiné dělení ještě): complete/exhaustive, heuristic, random, evolutionary
- **Metody redukce počtu instancí (numerosity reduction)** - např. u kNN je výhodná (paměť a výpoč.)
  - Voronoi diagram - rozdělení na oblasti, které pokrývají dané instance
  - Proximity grafy (poskytují různé definice souseda): nearest neighbour graf, minimum spanning tree, RNG (radiální sféry), Gabriel (2 body, kružnice), Delaunay (3 body, kružnice)
  - Skládá se z editování (odstranění šumu, aby vznikly homogenní clusteru) a konzdenzace (vybírá podmnožinu či generuje novou množinu tak, aby nedošlo k výrazné změně klasif. přesnosti)
  - Metody editace: Wilsonovo editování (odstraň body jiné třídy než většina sousedů), Multi-edit (opakovaný Wilson na náhodné části)
  - Metody kondenzace:
    - Neadaptivní (vybírá podmnožinu, zaměřuje se na instance na hranicích): CNN (začíná s jednou instancí v subsetu), RNN (další redukce CNN), IB3 (podobné CNN, confidence), DROP3
    - Adaptivní (generuje nové instance): Prototype (vytvoří nové strategické body), Chen (centroidy), RSP (Chen s balanc. tříd)
  - Jak velký vzorek - závislé na počtu dost. dat, distribuci, hustotě, počtu proměnných, je dobré vzít třeba 10 instancí a udělat si křivku s distribucí a pak přidat dalších 10, pokud je stejná tak už nepřidávat, jinak přidat ještě

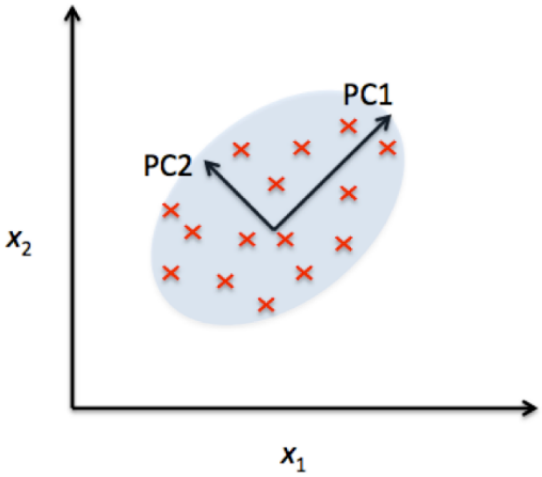
6.2 Algoritmy pro nahrazování chybějících hodnot. Detekce a ošetření odlehlých hodnot. Balancování a transformace dat.

- Metody redukce dat, které zároveň balancují data (takže patří asi jak do 1. tak do 2. ohruhu)
  - Under-sampling metody - náh. elimin. instancí z major. třídy, může zahodit důležitá data
    - Tomek links - odstraňuje šum a instance na hranicích
    - CNN - vybírá body blízko hranice mezi třídami, nejdřív 1 sample pak eventuelně další přídává

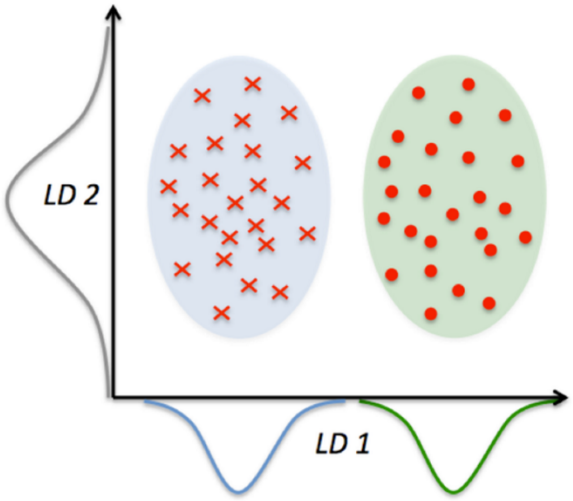
        * One-sided výběr = Tomek links + CNN
        * CNN + Tomek links - nalezení Tomkových spojení je drahé, tudíž lepší dělat na reduk. setu
        * Neighbourhood cleaning rule - odstranění instancí major. třídy
    - Over-sampling metody (náh. replikace instancí z minor. třídy, může zvýšit overfitting): Smote - vytvoří nové body z minor. třídy pomocí interpolace mezi body z minor. třídy
    - Kombinace
        * Smote + Tomek links - Smote může vytvořit až moc umělých minor. bodů hluboko v prostoru major. bodů, Tomek data vyčistí, ale musí se použít na obě třídy, ne jen major.
        * Smote + ENN - ENN odstraní body jejichž třída je odlišná alespon v 2/3 NN
        * Nejlepší pro použití na datech s malým počtem positive bodů.
- **Chybějící hodnoty** - těžko poznat jestli je hodnota chybějící nebo chybná, lze: odstranit záznamy, nahradit minimem/průměrem, nahradit pomocí kNN (nejlepší).
- Konverze dat: binary->numeric, ordered->numeric (A-1,B-1.5), nominal->few_values (žlutá-100), nominal->many_values (vyber nejčastější, zbytek spoj dohromady)
- Diskretizace dat (binning) - ve fázi čištění dat, užitečné pro generování přehledu dat
    - Binning: equal-width (stejně velké intervaly), equal-height (nebo někdy depth nebo frequency, přibližně stejný počet instancí v každém intervalu
    - Outliers co dělat: nic, horní/spodní meze, nech na binningu, použití shlukové analýzy
    - Shluk. analýza - pro nalezení outlierů, např. K-means, heirarchické shlukování
- Výběr polí - zahodit pole s malou či žádnou variabilitou, zjistit počet unik. hodnot
- Falešné prediktory (inf. leakers) - pole co korelují z cílovým chováním, v DB bez dat událostí, např. falešný prediktor šance že student uspěje v předmětu je jeho závěrečná známka
- Selekce nejrelev. polí - nechat jen ty s největší přesností
- **Balancování dat** - musí být vyvážené množství instanci z jedn. tříd, pro více tříd se to jmenuje Stratified sampling
- **Transformace dat** - smoothing (odstraní šum), feature construction (vytvoř. nových příznaků ze zadaných), transf. na unif. distribuci, normalizace (min-max, z-score, decimal scaling, softmax scaling)

6.3 Lineární projekce dat do prostoru o méně dimenzích (PCA, LDA), nelineární metody redukce dimensionality (Sammonova projekce).

- PCA se snaží zachovat varianci v datech (= information) – hledá dimenze s největší variancí
    - how it works: calculating the covariance matrix of the data (easy peasy) → calculating the eigenvectors and eigenvalues of the matrix (lemon squeezy) → sorting these eigenvectors according to their eigenvalues
- LDA se snaží dimenze redukovat tak, aby se zachovala co největší odlišnost tříd
- PCA je unsupervised, LDA supervised (taky se používá pro klasifikaci)
- Sammonova projekce
    - snaží se zachovat vzdálenost jednotlivých bodů od sebe
    - počítá Sammonův stress (error), což je error funkce, která kouká na vzdálenost bodů v původním prostoru a na jejich vzdálenost po projekci do menšího prostoru
    - tahle funkce je minimalizována třeba pomocí gradient descent



Obrázek 3: PCA



Obrázek 4: LDA

7 ADM

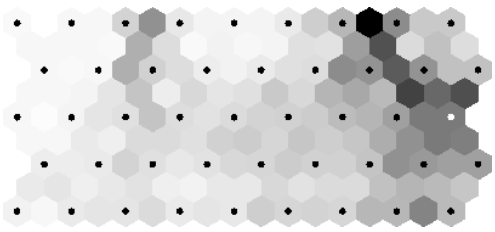
7.1 Metody klasifikace a regrese. Algoritmus nejbližších sousedů. SVM klasifikátory. Rozhodovací stromy a jejich varianty (Random forest, Boosted trees).

- Metodiky: SEMMA, CRISP-DM
- Modely: predikční (klasifikace, regrese), popisné (segmentace, shlukování)
- Šum, bias, variance
- Fáze učení (trénování) a fáze použití (vybavování)
- Chyba modelu: klasifikace: procento nesprávných klasifikací, regrese: součet čtverců odchylek, průměr čtverců, odmocnina průměru
- Metrika = vzdálenost vzorů – euklid, manhattan, cebysev
- Cross-validation: k-fold, leave-p-out
- kNN – velmi populární a pomalé, kruciální k, použitelné i pro regresi
- Decision trees
  - Split criteria: information gain (change in information entropy), information gain ratio (normalised), chi^2 test, gini index
  - When to stop: only one class in the set, too few instances in the set, no more information gain
  - Advantages: immediate insight, fast, easy to alter
  - Disadvantages: bad for continuous variables, limited accuracy, may be unstable
  - Random forest: N random subsets (with repetition) for N trees → e.g. majority voting
  - Boosted trees (xgboost) - malé, jednoduché stromy jsou pospojovány za sebou. První fituje samotná data, ten druhý za ním fituje residua = rozdíl predikcí prvního stromu a true hodnot, ...
- SVM – perceptron s lineární výstupní funkcí: linear, multiclass (one-against-all, one-against-one), nonlinear (jadernou transformací se přejde do vyšší dimenze a tam už se jede lineárně; poté zpět o dimenzi níž). Použití: filtrace spamu, obsahové filtrování (videa, obrázky) – doporučování, detekce malware nebo síťových útoků
- Ensemble metody: Bagging (Bootstrap Aggregating), boosting, stacking

7.2 Lineární, polynomiální a logistická regrese. Klasifikace pomocí perceptronu. Vícevrstvá perceptronová síť a její učení.

- Lineární regrese = aproximace pozorovaných hodnot daným typem funkce, parametry funkce lze určit metodou nejmenších čtverců (ta se v podstatě rovná regresi)
- Gro metody: Funkce $f(x) = \beta_0 + \beta_1 x + \epsilon$. Chceme minimalizovat sumu errorů $\sum_i (f(x_i) - y_i)^2$, tj. hledáme optimální bety. Rozjedeme derivace a vyjde nám $\beta_1 = \frac{S_{xy}}{S_{xx}}, b_0 = \overline{y} - \beta_1 \overline{x}$. Výběrová kovariance $S_{xy} = \frac{\sum_i (x_i - \overline{x})(y_i - \overline{y})}{n-1}$ a výběrový rozptyl $S_{xx} = \frac{\sum_i (x_i - \overline{x})^2)}{n-1}$.
- Polynomiální regrese: $f(x) = \beta_0 + \beta_1 x + \beta_2 x^2$
- Obecná metoda nejmenších čtverců: $y = \mathbf{X}\beta, \beta = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T y$
- Logistická regrese = taková lineární regrese pro klasifikační problémy. Docela rozumně vysvětleno zde. Používá logistickou distribuci místo normální, jsou hodně podobný, ale logistická funguje líp na problémy s výstupem $[0, 1]$.
- Nelineární regrese – solved pomocí Gauss-Newtonovy metody.
- Perceptron – potřeba lineární separability klasifikačního problému. Jsou to klasický jednoduchý neurony: vážený vstupy jsou sečteny, k tomu přidán bias a šup s tím do aktivační funkce – Heavisideova fce = 0 pro vstup menší než 0; 1 pro vstup většírovno 0. Učení probíhá pomocí postupnýho upravování vah: $\Delta w = \epsilon(\delta - f(x))x$. Kroneckerova delta je správný výstup, $f(x)$ je výstup perceptronu. ϵ je rychlost učení. Věta o konvergenci učícího algoritmu perceptronu říká, že to funguje.
- Vícevrstvý perceptron: sigmoidální funkce místo Heavisideovy (sigmoid, hyperbolický tangent). Učení pomocí backpropagace. Prej existujou i lepšejší metody dneska: metody sdružených gradientů, různých quasinewtonovských metod, a především různých variant Levenberg-Marquardtovy metody.

7.3 Shluková analýza (algoritmy K-středů, hierarchické shlukování, neuronový plyn, SOM).

- Shluková analýza = unsupervised. Optimalizační problém, neznámé: počet shluků, přiřazení dat do shluků.
- Hieararchické shlukování = dendrogram. Vyhodnocení vzdálenosti shluků: metoda nejbližšího/nejvzdálenějšího souseda, centroidní metoda, metoda průměrné vazby, Wardova metoda
- CPCC (cophenetic correlation coefficient) – říká, jestli data opravdu obsahují shluky (čím větší tím lepší). Je to normovaná kovariance vzdáleností v původním prostoru a v dendrogrmau.
- k-means
- Silhouette – zjišťuje, jak dobře jsou data klastrována. $s(i) = \frac{b(i) - a(i)}{\max a(i), b(i)}$, $a(i)$ je průměrná vzdálenost samplu i od samplů ze stejného klastru. $b(i)$ je průměrná vzdálenost samplu i od samplů ze ostatních klastrů. s může být -1 až 1, čím větší, tím je ten sample líp zařazen.
- SOM = neurony jsou rozesety ve 2D mapě a postupně se posouvají (competitive learning - vítězný neuron bere vše) tak, aby výsledná mapa co nejlépe překrývala trénovací data. Tj. v okolí (třeba gausovském) každého samplu se najde nejbližší neuron a ten se posune ještě blíž.
- 
- U-matice = zobrazuje vzdálenosti mezi jednotlivými neurony v SOM
- 

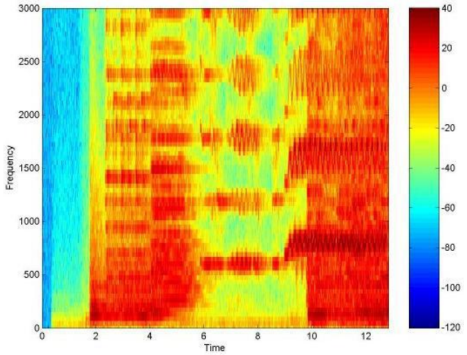
- Tečky jsou neurony. Tmavý políčka značí velkou vzdálenost mezi nima, světlý políčka malou vzdálenost. Neurony vpravo nahoře jsou tedy docela oddělenej klastr od ostatních.
- P-matice = odráží hustotu dat. Doplňuje informace z U-matice.
- U*-matice = kombinace U-matice a P-matice. Vzdálenosti mezi sousedními neurony (neurony a a b v mřížce) jsou vypočítány z U-matice a jsou váženy hustotou vektorů kolem neuronu a.
- Nevýhody těchto matic: nejsou intuitivně interpretovatelné, při novém naučení na stejných datech můžou vypadat jinak. Lepšejší: PCA, LDA, Sammonova projekce
- Neuronový plyn = podobný SOM, okolí neudává původní prostor dat, ale prostor neuronů.

8 VMW

8.1 Textové a bag-of-words modely pro content-based retrieval. Podobnostní model vyhledávání, podobnostní míry a dotazy.

- Bag-of-words model - set slov, u každého četnost, běžně se používá u klasifikace, také pro feature generation
- N-gram model - to samé jako bag-of-words model akorát pro n-tice slov
- Bag of features (bag of visual words) - nemáme anotaci obrázku, vyvoříme ji (visual words místo slov) pomocí jeho obsahu, features jsou podobrázky, visual vocabulary (vezmi všechny feature vektory, cluster them, použij centroidy jako visual words)
- Information retrieval pojmy: binární podobnost, způsoby vyhledávání (dotazování, browsing, filtrování), kvalita vyhledávání (měřítka kvality: přesnost, úplnost), ground distance, deskriptor, stemming
- Podobnostní model vyhledávání - určuje způsob vyhodnocení dotazu nad DB dat, cílem dostat co nejvíce dokum., které jsou relevantní, eventuelně i seřazené podle relevance
  - Vyhl. v textových dok. - extrakce termů, odfiltrují se zbytečné věci (např.the/a), každý dok. je pak reprentovám množinou termů: kolekce (množ. všech dokumentů a jejich id), slovník (množ. všech termů a jejich id ze všech dokumentů v kolekci), dokument (množina termů v dokumentu)
    - Modely: Booleovský (reprez. bitovým vektorem), Vektorový (reprez. vektorem vysoké dimenze)
  - Vyhl. multimedií - anotace dat a vyhl. v metadatech, přímé hledání v obsahu (extrakce nějakých vlastností množina vlastností potom tvoří deskriptor daného objektu)
- Podobnostní míry - říká nám jak moc jsou 2 objekty podobné
  - Metrika - musí splňovát tyto vlastnosti: reflexivita, nenegativita, symetrie, trojúh. nerovnost
  - Nemetrika - každá podobnostní funkce, která nesplňuje postuláty metricity
  - Vektorové vzdálenosti: L_p vzdálenost (Minkowskiho, L_1 Manhattonská, L_2 Euklidovská, L_∞ Čebyševova), zlomková L_p, kosinová/uúhlová
  - Histogramové vzdálenosti: vážená Eukl. vzd. wL_2 (každá dimenze má svou váhu), statistické (Kullback-Leibler divergence, Jeffrey divergence, Quadratic Form Distance, Earth-mover's vzd.)
  - Další vlastnosti vzdálostí: skládání metrik, skládání nemetrik, vektorové prostory
  - Vzdálenost řad: dynamic time warping distance (umí natáhnout jednu řadu na délku druhé)
  - Vzdálenost řetězců: editační vzd., Hammingova vzd., Longest Common subsequence
  - Vzdálenost množin: Jaccardova vzd., Hausdorffova vzd.
  - Vzdálenost grafů: tree edit vzdálenost
  - Kombinované vzdálenosti: deskriptor se skádá z více sub-deskriptorů (každý má vlasntí podobnostní prostor), pro každý dotaz se vyhodnocuje jiná sada sub-deskriptorů
- Podobnostní dotazy - způsob zadání dotazu systému, který podle konkrétního modelu upravuje množinu objektů, které vrací jako vyhovující dotazu
  - Jednoduché dotazy: range query, kNN dotaz, RkNN (Reverze kNN)
    - Pokročilé dotazy - ve velkých DB může být mnoho duplikátů a kNN ztrácí svou sílu, např. DkNN (Distinct kNN), víceobjektové dotazy
  - Dotazovací jazyky - rozšíření SQL o nové predikáty do klauzulí WHERE a FROM (např. range, kNN...)

8.2 Extrakce vlastností z multimédií pro potřeby vyhledávání - techniky pro obrázky, audio, video, geometrie. MPEG-7 deskriptory, lokální obrazové deskriptory (SIFT, SURF).

- Obrázky
  - Vlastnosti: lokální/globální, low level (barva, odstín)/high level (objekty)
  - MPEG-7 deskriptory: rozhraní pro popis multimedií s cílem efektivního vyhledávání (formát založen na XML)
    - Deskriptory (D): založené na katalozích (název, autor, práva, popis), sémantice (kdo, co, kdy a kde – informace o objektech a událostech), syntaxi (pro CBR, barva obrazu, tón zvuku) a technologii (formát, velikost, vzorkovací frekvence)
    - Popisová schémata (DS) popisují strukturu a sémantiku vztahů mezi komponentami D nebo DS – typ média, jeho původ, možnosti použití, strukturální vlastnosti nebo libovolný text
    - Data Description Language = všechno to definuje
    - Deskriptory barvy (deskriptor pro barevný prostor – např. EGB, pro kvantizaci barev, pro dominantní barvu, scalable colour D, colour layout D, colour structure D)
    - Deskriptory textury (homogeneous texture D, texture browsing D, edge histogram D)
  - Local image features
    - Detekce hran a shluků dat (hrana = změna jasu)
    - vyhledávání určitých významných bodů zájmu (identifikovány jako lokální maxima nebo minima z Difference of Gaussians (DoG) v obrázku o různých měřítcích)
    - SIFT (Scale-invariant feature transform) = Automatické nalezení korespondencí mezi dvojicí obrázků, použití: porovnávání nebo rozpoznávání obrázků
    - SURF (Speeded up robust features) = Body zájmu nejsou založeny na histogramu (na rozdíl od SIFTU) ale na součtu intensit. Mezi hlavní rozdíly, které přispívají k výraznému zrychlení, je nahrazení konvolučního jádra podle derivace gaussovy funkce jádry založenými na funkci obdélníkové.
    - Vlastnosti tvarů či shoda tvarů (k-means)
- Video
  - Jednotlivé framy (analýza neprobíhá), shot (záběr, zajímavé pro přechody mezi záběry – poměr změny hran), scéna (množina souvisejících záběrů), celé video – *vlastně vůbec nechápu proč tu tenhle seznam vůbec je, wtf*
  - Keyframes = Sled framu (obrazku), ktere nejak popisuji video = hodi se pro tvorbu deskriptoru. lze na ne pouzit image retrieval. 2 typy: highlighting (ty nejpodstatnejsi), summarizing (celkove)
  - MPEG-7 & video – GoF/GoP deskriptor (group of frames/pictures, založeny na barvách a histogramech, I-frame, P-frame, B-frame), lokalizační deskriptor (chci okno, dám tam čtverec a doufám, že to najde), motion deskriptor (motion activity, camera motion, parametric motion, motion trajectory)
- Audio, hudba, melodie
  - Fourierova transformace – z časové domény do frekvenční (užitečné např. pro komprimaci zvuku – mp3)
  - Spektrogram – vizuální reprezentace zvukového signálu
  - 

  - MPEG-7 – 17 nízkoúrovňových a 5? vysokoúrovňových deskriptorů.

    - Základní spektrální analýzy: AudioWaveform (min max hodnoty amplitudy v time frames), AudioPower (okamžitý vyhlazený výkon zvukového signálu), AudioSpectrumEnvelope (frekvenční spektrum), AudioSpectrumCentroid (těžiště frekvenčního spektra), AudioSpectrumSpread, AudioSpectrumFlatness
    - Základní parametry signálu: AudioHarmonicity, HarmonicRatio (míra podílu harmonických složek ve spektru), UpperLimitOfHarmonicity (odhad frekvence, po kterém spektrum již nemá žádnou harmonickou strukturu), AudioFundamentalFrequency
    - Časové deskriptory: ADSR zvlnění (attack, decay, sustain, release), LogAttackTime, TemporalCentroid, HarmonicSpectralCentroid, HarmonicSpectralDeviation, HarmonicSpectralSpread, HarmonicSpectralVariation, SpectralCentroid, Mel-FrequencyCepstrumCoefficients (vynikající funkce vektoru představujícího lidský hlas a hudební signály, slouží především k rozpoznávání řeči a hudebnímu vyhledávání, žánrová klasifikace, audio podobnosti)

- Vyhledávání 2D tvarů a 3D modelů
  - MPEG-7 & tvary – region shape desk, contour shape desk, uzavřené/otevřené polygony, porovnávání tvaru DTW, porovnávání tvaru pomocí nejdelší společné sekvence LCSS
  - 3D model – transformation to 2D shape retrieval, skeletonizace, 3D interest points, porovnávání siluety

9 PDB

9.1 Vyhodnocování a optimalizace SQL.

- Stavy v diagramu transakce: aktivní (na počátku), částečně potvrzený (po provedení poslední operace transakce), chybný (nelze pokračovat v normálním průběhu), zrušený (stav nastane po skončení operace rollback), potvrzený (po úspěšném vykonání commit)
- Způsoby uložení:
  - Interní struktura B-stromového indexu - listy jsou zřetězené, přístup na disk je při každém skoku: z rodiče na potomka (proto je mělký), z listu (kde je adresa dat. segmentu) k samotnému řádku
  - Indexově organizovaná tabulka - podobné B-stromovému indexu, ale v listech jsou přímo data, ne jen index
  - Tabulka ve shluku (cluster) - náhrada spojení přes klíč (v dat. bloku je záznam z tabulky a hned pod ním záznamy z jiné tabulky), které k němu patří
- Fáze zpracování SQL příkazu: 1) převod do vnitřní formy 2) konverze do kanonického tvaru 3) optimalizace (a. vyčíslení alternativních prováděcích plánů b. výběr toho s nejnižší cenou c. transformace dotazu) 4) plán vyhodnocení 5) generování kódu
- Výpočet dotazu - podle prováděcí metody, opírá se o statistiky nad tabulkami a indexy, počítá se I/O
- Statistiky tabulek (vše v tabulce R): počet řádků N_R, počet různých hodnot atributu A je $V(A,R)$, počet stránek v tabulce (kB) $P_R = N_R/B_R$, blok faktor (prům. počet řádek v jedné stránce) B_R, počet bloků co se vejde do paměti M
- Statistiky indexů: faktor větvení (počet přímých potomků, $50-100$) $f(A,R)$, hloubka indexového stromu $(2-3)$ $I(A,R)$, počet bloků v listu $p(A,R)$, počet záznamů v listu $B_i(A,R)$
- Různé metody selekcí pro:
  - Tabulku: 1) unikátní atribut - průměrně $P_{emp}/2$, 2) není unikátní - full table scan
  - B-stromový index (data v blocích nejdou za sebou jako indexy, pro každý řádek nutno přitoupit do datového bloku zvlášť, počet hledaných záznamů N_{hz}): 1) $N_{hz} = 1$, $cena = I(A,R) + 1$, 2) $cena = I(A,R) + N_{hz}/B_i + N_{hz}$
  - Cluster (clusterový index, podobné B-stromu, data v dat. blocích za sebou jako indexy B-stromu): 1) $cena = I(A,R) + 1$, 2) $cena = I(A,R) + N_{hz}/B_i + N_{hz}/B_E$
  - Indexem organizovaná tabulka (data fyzicky seřazena přímo v B-stromu, tozn. že odpadá cesta do dat. bloku): 1) $cena = I(A,R)$, 2) $cena = I(A,R) + N_{hz}/B_i$
- Algoritmy spojení (počet bloků paměti M - 3 je minimum)

- Nested loops: 1) $M = 3$ platí $cena = P_{mensi} + P_{mensi} * P_{vetsi}$, 2) $M = 4$ platí $cena = P_{mensi} + (P_{mensi} - 1) * P_{vetsi}$, maximální počet M které efektivně využijem záleží na P_{mensi}
- Merge join (seřadíme první, seřadíme druhou, sekvenčně projdeme): $M = 3$ platí $cena = 2 * P_E * \log(P_E) + 2 * P_D * log(P_D) + P_E + P_D$, idealní takové M abychom dokázali seřadit větší relaci v 1 kroku
- Hash join - menší vnější, pomocí vnější vytvoří hash tabulku do větší, tato tabulka se načte do paměti a pak se prochází větší tabulka a k řádkům se hledají odpovídající řádky

- Prováděcí plán - při vyhodnocování jsou důležité různé statistiky (vel. tabulky, vel. indexu, atd.)
  - Podle statistik se DBMS rozhodne při vytváření prováděcího plánu a ten se optimalizuje na nejmenší počet I/O operací
- Metody zpracování: 1) On-the-fly (pipeline, rovnou se vyhodnocuje) 2) Materialization (ukládání mezivýsledků)
- Vytvořené plány jsou ukládány a kdyžtak znovu použity s jinými parametry třeba.

9.2 Databázové modely a dotazovací jazyky.

- Databázový model = typ datového modelu, který určuje strukturu databáze a říká, jak tam data budou ukládána a organizována
- Relační vs objektová tabulka: řádky, sloupce, omezené množství datových typů, cizí klíče vs třídy s atributy a metodami, zapouzdřením, dědičností a abstraktními datovými typy
- BigData = buzzword. high Volume, Velocity, Variety
- NoSQL (not only sql) databáze:
  - Motivace: Klasické relační databáze jsou navrženy do OLTP systémů a dodržují ACID (Atomicity, Consistency, Isolation, Durability). NoSQL jsou BASE (Basically Available – většinu času dostupná, Soft state – stav systému se může změnit i bez inputu, Eventual consistency – systém se časem stane konzistentní, pokud teda nepřijde nový input)
  - Typická charakteristika: nerelační, distribuované, horizontálně škálovatelné (přidávání uzlů) open-source, schema-free, jednoduché API, big DATA
  - Klasifikace: široké sloupce/rodiny sloupců (BigTable, Hadoop/HBase, dokumentové úložiště (např. JSON, XML – MongoDB, Terrastore), key-value (LevelDB, HamsterDB), eventually consistent key value store (Amazon Dynamo, Voldemort), grafové db (Neo4J, HyperGraphDB)
  - XML databáze
    - schema-free/schema-based
    - ukládání: filesystem aka texťák (easy s existujícím API, ale pomalé a neefektivní), XML-enabled (aka naperu XML do relační db), native XML (využívá B-stromy)
    - jazyky pro dotazování: XPath, XQuery
  - SQL/XML: rozšíření SQL o XML, Oracle, IBM, i trochu MS SQL.
  - Sedna: schema-based, B-trees
  - Grafové db: uzly, hrany, bezindexová sousednost (uzly obsahují přímé odkazy na sousedy), využití = FACEBOOK, vazby jednosměrné/obousměrné, traverzování = průchod grafem umožňující navštívit všechny vrcholy (BFS, DFS)
  - MapReduce: map rozdělí data mezi nody, a reduce je pak spojí a třeba něco spočítá. Fault tolerant. Apache Hadoop.
  - RDF (Resource description framework): a framework for representing information in the Web. Resource: anything that is identifiable by IRI. Triple: Subject - Who, Predicate - Property or characteristic, Object - value of property. SPARQL – query language for RDF. Používá se pro semantic web
  - Key-Value db: Simplest NoSQL data store – hash table. Riak (keys in buckets, usage: HTTP), Redis

9.3 CAP teorém a NoSQL databáze.

- CAP = říká, že v distribuovaných systémech je možné dodržet jen dvě ze tří uvedených vlastností: Consistency (Every read receives the most recent write or an error), Availability (Every request receives a (non-error) response – without guarantee that it contains the most recent write), Partition Tolerance (The system continues to operate despite an arbitrary number of messages being dropped (or delayed) by the network between nodes).
- NoSQL (not only sql) databáze: o otázku výše

10 PIS

10.1 Architektury informačních systémů - distribuované systémy v architektuře client server a centralizované systémy s v třívrstvé architektuře s lehkým klientem, využití Cloud computingu.

- **Distribuované systémy v architektuře client server**
  - Distribuované systémy - distr. zpracování, distr. data
    - Výhody: sdílení komponent, specializace komp., rozšiřitelnost a škál.
    - Nevýhody: složitost, vysoké požadavky na síť
  - Klient-server model - popisuje komunikaci mezi konzumentem služeb (klient) a poskytovatelem služeb (server)
    - nezávislé par. procesy, na jendom či více počítačích, mají své informace u sebe, komunikují defin. protokolem
    - Evoluce: 1) 2-vrstvý model 2) 3-vrstvý model 3) N-vrstvý model
  - Gartner model distr. systému - presentation layer, processing layer, data layer (hodně zjednodušeně)
  - Komunikace v distr. systému: (synch., asynch.), (pull model, push model), (přímá, zprostředkovaná), (request/response, fire and forget), (přímo adresované, nepřímo adr.)
- **3-vrstvá arch.** - centralizované systémy
  - Vrstvy: prezentační (tenký klient, žádná aplikační logika), aplikační, datová
    - Aplikační a datová - na serveru - central., snadná škál.
    - Každá vrstva může být vyvíjena/měněna nezávisle.
    - Výhody: malý síť. provoz, při změně apl. logicky stačí akt. jen server, klient nemusí znát kde jsou data, sdílení přip. k DB, standardiz. SQL
- **Využití Cloud computingu - Architektura inform. systémů v cloudu**
  - Typy: Cloud Clients (mobile app, thin client, web browser), SaaS (email, virtual desktop, communication, games), PaaS (DB, web server), IaaS (virt. machines, storage, network, load balancers)
  - Business výhody: není třeba investovat peníze, je to hned, platím jen to co potřebuju, výkon mám kdy potřebuju
  - Technické výhody: auto-scaling, proactive-scaling (zátěž), lepší vývoj. cyklus a testovatelnost
  - Návrh: rozšiřitelnost, oddělení komponent, elastičnost (paralelní procesy), statická data udržovat u uživatele a naopak

10.2 Práce se vstupními a výstupními daty v informačních systémech - konsolidace, normalizace, agregace, Key Performance Indicators (výkonové ukazatele).

- Konsolidace = sběr a integrace dat z více zdrojů do jednoho místa (odstraňuje duplikace, snižuje náklady na údržbu)
- Normalizace = proces vytváření malých, stabilních a flexibilních datových struktur z komplexních skupin dat (takový to 1NF, 2NF, 3NF, BCNF)
  - Optimalizace tabulek, klíčů, sloupců a vzájemných vztahů v relačních databázích

  - Výhody: integrita dat (updatem nic nerozbiju), optimalizované dotazy, rychlejší řazení a update, zlepšení řízení souběžného přístupu
  - OLTP (OnLine Transaction processing) – INSERT, UPDATE, DELETE – vyžaduje normalizace vs. OLAP (.. analytical ..) – jen análýza, nevyžaduje normalizaci.
  - Normalizovaná data je poté možno zvalidovat (odstranění kódování a převod do jedné znakové sady)
- Agregace (seskupování) = OLAP. Komponenta BI – dolování a analýza dat, např. statistické analýzy.
- Key Performance Indicators = pomůcka pro měření výkonnosti (úspěšnosti) organizace
  - Skládá se z několika dimenzí (oblastí), každá má své metriky
  - Proces identifikace KPI: 1. definovat byznys procesy a jejich požadavky. 2. měřit kvantitativní a kvalitativní ukazatele těchto procesů a porovnávat výsledky se stanovenými cíli. 3. prozkoumávat možné změny v procesech, které umožní dosažení cílů.
  - Balanced Scorecard = Metoda (nástroj) v managementu vytvářející (reálnou) vazbu mezi strategií a operativními činnostmi s důrazem na měření výkonu. Dimenze: finance (obrat, tržby, zisk na zakázku, produktivita na pracovníka), zákazníci (podíl na trhu, spokojenost zákazníka), interní procesy (počet prodaných produktů, doba dodávky), učení se a růst (počet seniorních specialistů)
- Reporting = Činnost spojená s dotazováním se do databází pomocí jejich standardních rozhraní (SQL dotazy). OLAP.
- Vizualizace = Obrázek za 1000 slov – vytváření, upravování, zkoumání grafíků. Dashboard = tachometr, aktuální informace (např. nejdůležitější KPI) vs Scoreboard = navigace, celkový pohled.

11 MVI

11.1 Evoluční algoritmy, schémata, diversifikace populace.

- **Typy evolučních algoritmů**: GA, GP, EP, neuroevoluce.
- Pojmy: individuál (jedinec), chromozom, fitness, gen, genotyp, fenotyp.
- Typy reprezentace: reálná čísla, stromy, bin. řetězec, obrázek etc.
- Postup:
  1. Inicializace populace - náhodná nebo informovaná (seeding).
  2. Selekce: ruleta, turnaj, rank.
  3. Křížení: 1-bodové, 2-bodové, cut&splice, uniformní.
  4. Mutace: nahrazení, prohození, inverze, smazání, etc.
  5. Vyhodnocení fitness pomocí fitness funkce
- **Schémata** - templaty, skládající se z fixních symbolů (0,1) a hvězdiček, vlastnosti:
  - délka d(S) - vzdálenost mezi prvním a posledním fixním symbolem (pro křížení)
  - řád o(S) - počet fixních symbolů (pro mutaci)
  - fitness f(S) - avg. fitness spočítané přes všechny řetězce schématu
  - m(S,t) - počet jedinců patřících schématu S v čase t
  - $f_s(t)$ - avg. fitness jedinců patřících do schématu v čase t
  - f(t) - avg. fitness přes celou pop. (i ty řetězce nepatřící do schématu)
  - pro selekci platí: $m(S, t+1) = m(S, t) * (f_s(t)/f(t))$
  - pro křížení platí: pravd. že schéma S (počet všech symbolů S je l) přežije křížení je $p_s(S) >= 1 - p_c * (d(S)/(l-1))$
  - Pravd. že přežije mutaci je $p_s(S) = (1-p_m)^{o(S)}$., při $p_m \ll 1$ platí $p_s(S) \approx 1 - p_m o(S)$
  - Celková pravděpodobnost přežití je $m(S, t+n) >= m(S, t) * (f_s(t)/f(t))^n * [1 - p_c * (d(S)/(l-1)) - p_m * o(S)]^n$
- **Diverzifikace populace** - používají se niching methods, mimo uvedené metody lze například použít i ostrovy.

- deterministic crowding = žádná selekce, náhodní rodiče do dvojic, mají 2 děti a podobnější dvojice rodič-dítě si dají turnaj a vyhraje ten lepší
- restricted competition = mezi všema dvojicema se při selekci zkusí kdo má větší fitness (když jsou dostatečně podobní) a tomu kdo prohrál se nastaví fitness na nulu
- fitness sharing - sníží se fitness každému jedinci na základě počtu téměř stejných jedinců k němu v populaci, je zvolen nějaký threshold, dá se použít Hammingova vzd. či Eukleidova

11.2 Evoluce neuronových sítí a rozhodovacích stromů.

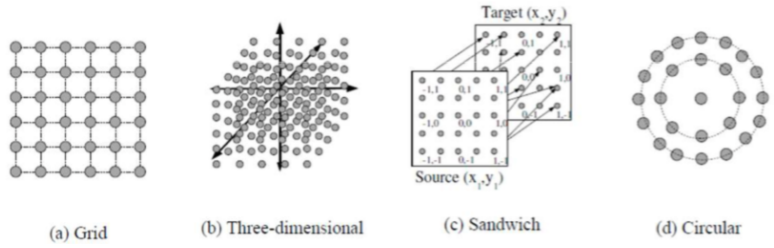
- **Evoluce neuronových sítí** - když nevíme jak by měla NN vypadat, je možné šlechtit strukturu i parametry
- Šlechtění struktury - změny spojení mezi: neurony, neurony a inputy, neurony a outputy
  - Spojení mohou být: odebrány, přídány, změněny.
  - Začíná se s malou náhodně spojenou sítí, postupně se přídávají neurony.
  - Zakódování: fixní nebo proměnná délka genomu, záleží na parametrech NN, např. const/var počet inputů neuronu
    - Binární kódování nevhodné, spíše objekt/struktura.
    - U každého neuronu poznačeno: s kým má spojení, jestli má input, jestli má output a jaký, jeho id.
  - Operátory: selekce (klasické metody), křížení (musí produkovat validní potomky), mutace (např. přidej spojení, odstraň, změň, eventuelně přidej input/output)
- Šlechtění parametrů neuronu - váhy inputů, aktivační funkce, threshold - běžně se šlechtí jejich konstanty
  - Zakódování - u každého neuronu je držen: vektor vah inputů, aktivační funkce (případně její parametry), threshold.
  - Operátory - selekce (ty samé metody), křížení (důležité aby se hodnoty nedostaly mimo nějaký rozsah třeba), mutace (např. změň input vektor, clear input vektor=nastav nuly, změň threshold, změň koef. aktivační funkce).
- Šlechtění gramatiky, která popisuje kontrukci NN
  - Nepracuje se stromy, ale s binárními/integerovými řetězci, ty jsou přeloženy do posloupností integerů (codons)
  - Genotyp-fenotyp mapování - každý codon specifikuje produkční pravidlo, které aplikuje pro daný neterminál: choice = codon MOD number_of_rules, mapování končí až jsou všechny neterminály rozexpandovány.
  - Díky gramatice mohou být vygenerovány pouze syntakticky korektní programy.
- **Evoluce rozhodovacích stromů** - podobně evoluci NN, evoluce gramatiky je častěji používaná
  - Každý strom je složen terminálů (real, integer, logické konstanty, akce) a neterminálů (aritmetické operátory, algebraické funkce, logické funkce, podmínky...).
  - Křížení: prohození podstromů jako v symb. reg.
  - Mutace: jako v symb. reg., např. nějaký podstrom je nahrazen novým random podstromem.
  - Mohou se použít i jiné operátory: permutation, editing, encapsulation, decimation.

11.3 HyperNEAT, Novelty Search.

- NEAT (NeuroEvolution of Augmenting Topologies) - šlechtí strukturu a váhy NN pomocí EA
  - Začíná v minimální formě.
  - Zakódování: 1) tabulka nodů (u každého je poznačeno v které vrstvě leží) 2) tabulka spojení (z jakého nodu, do jakého nodu, váha, enabled/disabled, inovační číslo).
  - Značkování (=historical markings, =inovation numbers) - když se objeví nové spojení (pomocí mutace), glob. inov. číslo se inkrementuje a přiřadí se tomu spojení, je použito k identifikaci stejných struktur v jedincích během křižení.

  - Mutace - vah, přidání spojení, přidání nodu; hrany (spojení) se nemažou, kdyžtak se dají na disabled.
  - Křížení - díky značkování bez složité topologické analýzy.
    * Geny které nepasují jsou buď rozdělené (disjoint) nebo nadbytečné (excess), záleží na tom jestli jsou v nebo mimo rozmězí inovačních čísel druhého rodiče.
    * Geny se stejnýma in. číslama jsou zarovnány a náhodně zvoleny do potomka, geny které nematchují jsou vzaté z více fit rodiče, disabled geny mají 25% šanci na reenable během křížení.
  - Speciation - vytvoření druhů kde jedinci soutěží pouze s tím samým druhem, jsou tak chraněny topologické inovace a je zabráněno bloating of genomes.
    * Jsou rozděleny do druhu na základě topologické podobnosti kdy se vypočte threshold.
    * Typy: pevný threshold, dynamic compatibility threshold (zvolen počet druhů), fitness sharing (každý druh bude mít jen omezeně jeidnců)

- **HyperNEAT** (Hypercube-based NEAT) - modifikace NEATu která reflektuje geometrii z input senzorů na neurony (symetrie, opakovaní, periodicita), těžko se to optimalizuje standardní cestou, kdy váhy nejsou závislé na pozici neuronu
  - Také klasické metody učení pro NN (e.g. backpropragation) nerespektují geometrický řád inputů, centra v lidském mozku ano.
  - Základem je funkce, která popisuje váhy jednotlivých spojení neuronů: $f((x_i, y_i), (x_j, y_j)) = w_{ij}$, tato funkce se šlechtí pomocí NEATu a struktura NN je šlechtěna separátním NEATem (který používá předchozí funkce k získání vah). Parametry funkce jsou 2D souřadnice neuronů.
- CPPN (Compositional Pattern Producing Networks) - je to NN s různými aktivačními funkcemi a neomezenými spojeními
  - Váhy sítě jsou vypočteny za pomocí souřadnic neuronu.
  - Substrate - seznam souřadnic neuronu společně s možnými spojeními mezi nimi (struktura sítě viz 5).

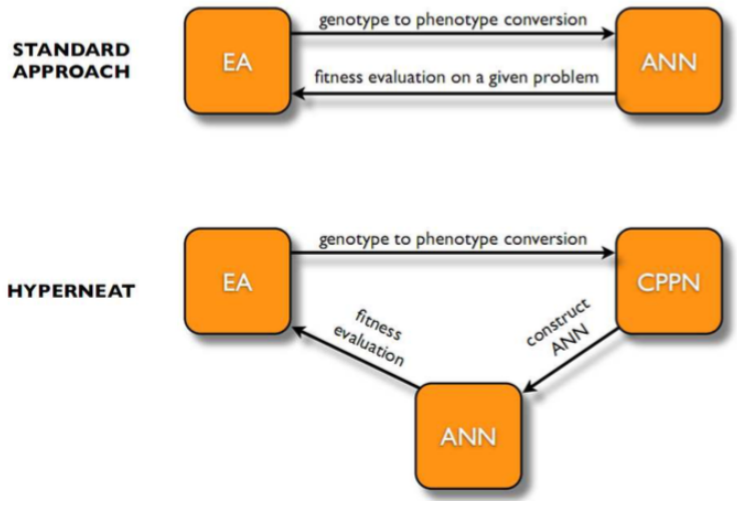


Obrázek 5: Typy struktur.

  - HyperNEAT vs standard approach viz 6
- **Novelty search** - bezcílné hledání, kde jsou odměňováni jedinci za nové objevy, i když třeba nesouvisí s cílem či mu odporují.
  - Je moc lok. optim, těžko se zopakuje šlechtící proces stejně.
  - Hodně věci bylo objeveno když se hledalo něco jiného, jiné zase potřebovaly nějaké stepping stones to be invented first i když nebyly přímo souvislé, dá se navigovat maximálně pár stepping stones, ale těžko více.
  - Každá nová věc je potenciální stepping stone.

11.4 Optimalizace hejnem částic (PSO) a mravenčí kolonií (ACO).

- Metody Swarm Intelligence (SI), což je kolektivní chování decentralizaovaných, samoorganizujících systémů, které jsou postaveny na populaci jednoduchých agentů (mají omezené schopnosti), kteří lokálně interagují spolu a se svým prostředím
- **PSO** - hejno částic, které se pochybují na zákl. jednoduchého pravidla v search space.
  - Průběh (to v závorce je cyklus): inicializace pozic a rychlostí náhodně, ohodnoť hejno, (update rychlostí pomocí personal best a global best, aplikuj nové rychlosti na pozice, ohodnoť hejno)



Obrázek 6: HyperNEAT vs standard approach.

  - Update rychlosti částice i: $v_{i+1} = a * v_i + b * \alpha * (p_i - x_i) + c * \beta * (g - x_i)$, kde a, b, c jsou koeficienty voleny uživatelem, p_i je nej. známá pozice částice i, g je nej. pozice částice z celého hejna a x_i je pozice částice i. $\alpha, \beta \in (0, 1)$ – random.
  - Update pozice se provede přičtením rychlosti k akt. pozici.

- **ACO** - kolonie mravenců s limitovanými schopnostmi, založeno na jejich chování při hledání potravy
  - Společná pamět realizována **feromony**.
    - Mravenci vytváří fer. stopu cestou zpět, pokud najdou potravu.
    - Čím je silnejší stopa, tím spíš ji následují ostatní, a tím ji posilují.
    - Tato fer. stopa časem vyprchává.
  - Pseudokód: cyklus(generateSolutions - napr. spoj 2 města v TSP, moveAnts - vytvoř nové part. řešení, pheromoneUpdate).
  - Pravd. pohybu mravence k z A do B: $p_{AB}^k = \frac{t_{AB}^\alpha * n_{AB}^\beta}{\sum_C (t_{AC}^\alpha * n_{AC}^\beta)}$, kde t je množství feromonů na dané cestě, n je přitažlivost dané cesty (typicky 1/délkaCesty), $\alpha >= 0$ kontroluje vypařování feromonů (vliv t) a $\beta <= 1$ kontroluje vliv n.
  - Vylepšení: elitist, rank-based ant system, max-min ant system (limituje nejlepší řešení na max, aby se zkoušely i jiný, a nejhorší na min, aby se úplně nezahodilo).
  - ACO v TSP: rozmístí se mravenci do různých měst a pravděpodobnostně volí další města, navštívená si dají do tabu listu, až všichni najdou nějakou cestu tak se updatujou feromony podle toho jak krátkou cestu našli (čím kratší cesta, tím silnější feromony na ni umístí), takto se může opakovat kolikrát je libo.
  - Nevýhody: pomalejší konvergence než jiné heuristiky, pomalejší výkon pro TSP s více než 25 městy.

11.5 Ensemble metody (boosting, bagging, stacking). Využití evolučních algoritmů.

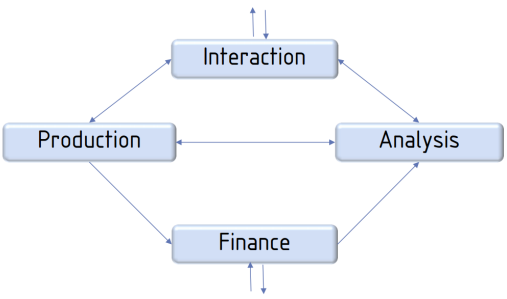
- **Ensemble metody** slouží k tomu abychom dostali lepší výsledky pomocí kombinace více modelů
  - každý model děla jiné chyby, dáním dohromady dostaneme lepší obrázek o datech
  - **Bias-Variance error decomposition**: $E = var_y\{y\} + bias^2 + var_{LS}\{y_{odhad}\}$
    - var_y je šum, *bias* je chyba prům. modelu proti optimálnímu, var_{LS} je jak rozdílné modely dostaneme z rozdílných trénovacích dat. Rozepsaně: $E = E_y\{(E_y\{y\} - y)^2\} + (E_y\{y\} - E_{LS}\{y_{odhad}\})^2 + E_{LS}\{(E_{LS}\{y_{odhad}\} - y_{odhad})^2\}$
  - Underfitting = modely mají nízký rozptyl a vysoký bias (velkou chybu oproti původním datům)
  - Overfitting = modely mají malý bias (fitujou přesně), ale vysoký rozptyl (jsou dost jiné pro jiná data)

- Ensemblovací algoritmy - klíčem k úspěchu je diverzita modelu, toho lze dosáhnout buď používáním jiných modelů nebo používáním jiných trénovacích dat pro každý model.
  - Jak fungují:
    1. Zvolí učicí algoritmy a vytvoří datasety pro ně.
    2. Naučí zákl. modely - učení jednoho modelu nemusí být nezávislé a inputy modelů mohou být závislé na outputech ostatních modelů.
    3. Pro daný neznámý vektor dostane output jednotlivých modelů a spočítá output ensemblu.
  - **Bagging**: data - náhodně vybraný subset s opakováním, učení - nezávislé, output - průměr.
  - **Boosting**: data - ty data co mají větší chybu na předch. modelech jsou spíš vybrány jako trén. data pro další model, učení - consecutive (po sobě jdoucí), output - vážený průměr založený na výkonu modelu.
  - **Stacking**: data - každý base model dostane stejná data a je udělán meta-dataset z jejich outputů, meta-dataset je input pro meta-model, učení - base model je nezávislý, meta-model je závislý na base modelech, output - výstup meta-modelu.
- Hierarchical ensemble - zobecnění ensemblovacího principu, jako base modely jsou použity jiné ensembly.
- **Využití evol. algoritmů**
  - NASA antena Yagi - šlechtily se umístění a délky jednotlivých komponent, které mohou významně ovlivnit výkonové charakterisitiky antény
  - NASA antena ST5 - genotyp je strom, který specifikuje konstrukci jednoho drátu v 3D, větvení ve stromu způsobuje větvení drátu, příkazy: forward(délka) - 0..3 potomci, rotate-x/y/z(úhel) - 1 potomek netemrinál, minmalizuje se nějaká fitness funkce (hraje tam roli zisk té antény)
  - Návrh obvodu - genetické programování použito pro zkombinování jak topologie tak sizingu (numerical component values) pro obvody, které duplikují patentované funkcionality
  - Geneticky vyšlechtěná řadící síť
  - Human competitive results: samořídící auta, intelligent assistants (Siri, Cortana), učení se hrát hru z obrazu hry (DeepMind, GoodAI)
  - Picbreeder - evolutionary art, generátor náhodných hezkých obrázků

12 EDW

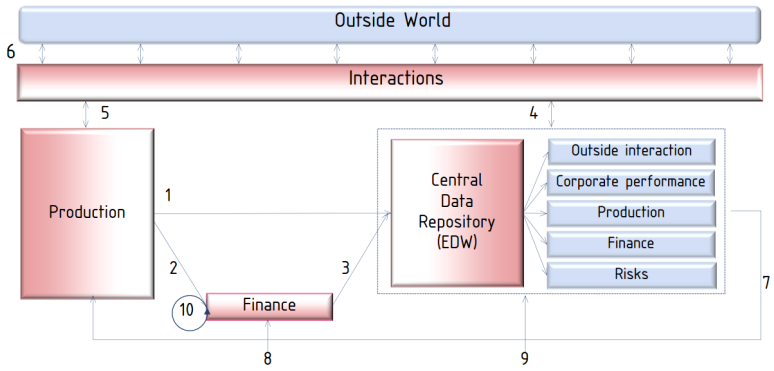
12.1 Business data model, datová architektura a modelování, extrakce dat a datová integrace.

- BI = It's an umbrella term that defines a broad range of applications, technologies and methodologies that support a user's access to and analysis of information for making decisions and managing performance.
- Model firmy se skládá z několika vrstev: Process & Organization (BI Governance), Information, Functional, Data, SW & Application, HW & Infrastructure. Každá vrstva vypadá jako na obr. 7.

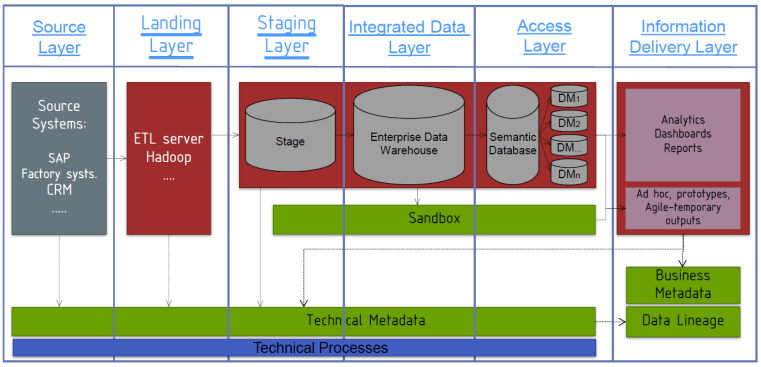


Obrázek 7: Jedna vrstva v modelu firmy.

- BIAF (Business Intelligence Architecture Framework) – obr. 8.
- Podmnožinou BIAF je BI architektura – obr. 9.



Obrázek 8: BIAF.



Obrázek 9: BI architektura.

- Datová integrace
  - Architektura – architektonický framework = Landscape pro logické uspořádání komponent na základě bussiness potřeb a technologických možností. Agnostické prostředí – nezávislé na konkrétních technologiích.
  - Design – identifikace implementace (výběr komponent), identifikace rozhraní a typu integrace (ident. implementace komponent), definice aplikací, úložišť (nástrojů), datových vrstev, kategorií a data integračních vzorů (pravidel; datové toky, load, ETL/ELT, integrační kroky).
  - Řešení – Vzory: source, landing, staging, integrated a access vrstev. Definice historizace dat (SCD).

12.2 MPP databáze pro datové sklady, Hadoop a discovery platformy, nástroje pro datovou integraci a reporting.

- Teradata is the best: hybrid storage, multi-temperature data, složena z několika cliques (pronounced, "kleek") = několik nodů a AMP (= Access Module Processor - asi cca procesor s HDD) - každej node může ke každýmu AMP uvnitř jednoho clique. Data jsou pomocí hashovací funkce rozdělena mezi různé AMP
- NoSQL: MongoDB - JSONy, BigTable - v jednom sloupečku několik dvojic proměnná:hodnota (Cassandra), Graph db - Neo4J
- Hadoop: data lake files, schema-on-read, raw data, scans, search, Spark, HDFS, technologist focused, freeform exploratory analytics, vice manualni
- Nástroje integrace: IBM InfoSphere DataStage
- Nástroje reportingu: MicroStrategy, IBM Cognos Insight, SAP BI, Microsoft BI, Tableau

12.3 Kontext a základní funkce Business Intelligence, analytické činnosti a rozhodovací procesy, uživatelské požadavky, roadmapa.

- BI = It's an umbrella term that defines a broad range of applications, technologies and methodologies that support a user's access to and analysis of information for making decisions and managing performance

- K čemu to je:
  - Automatizace poskytování dat pro podporu rozhodování
  - Zjednodušení tvorby informací ze získaných dat
  - Poskytnutí informací včas
  - Zajištění důvěryhodnosti poskytnutých informací
- Základní funkce:
  - Zpracovat uživatelské požadavky
  - Přibývají rychleji, než je možné je zpracovávát
  - Mnoho z nich vede na rozšiřování struktur datového skladu, building bloků a martů → "generují velké projekty"
  - Existuje mnoho malých požadavků, které mají vzájemné souvislosti – hlavně datové
  - Jejich seskupením („bundling") lze rovněž vytvořit definici „velkého projektu"
  - Jsou ale i malé změny, které takto zpracovat nejdou...
- Připravovat krátkodobé plány - Roadmapa
  - „Velké" projekty ("Releases")
  - Z „Velkých" projektů často „něco odpadává" do dalších Rel.
  - „Malé" projekty, ale pořád projekty ("Mini Releases")
- Připravovat vize/strategie - pro uživatele a pro BI
- Analytics:
  - Model driven: Theory → Hypothesis → Observation → Confirmation (Deductive reasoning (proof focused))
  - Discovery driven: Observation → Pattern → Hypothesis → Theory (Inductive reasoning (innovation focused))
  - Analytické činnnosti:
    - outside interaction
    - corporate performance
    - production
    - finance
    - risks

Zdroje

Edux, fit-wiki, wikipedia, ty internety, a tak různě...