Poslední návštěva: 02 úno 2011 12:32 am
Právě je 02 úno 2011 08:53 am
Svátek má Nela.
Nyní je lichý týden.

Termíny: (přidat, editovat, log)

IMP - Druhý opravný termín (3.2.)	Zbývá 1 dnů, 3 hodin a 6 minut
IMS - Druhý opravný termín (4.2.)	Zbývá 1 dnů, 23 hodin a 6 minut
Dotaznik SU FIT a SKAS (12.2.) - detaily	Zbývá 10 dnů, 15 hodin a 6 minut
IBP - Odevzdání BP (18.5.) - detaily	Zbývá 105 dnů, 9 hodin a 6 minut

Vyhledat témata bez odpovědí | Zobrazit aktivní témata
Zobrazit nepřečtené příspěvky | Zobrazit nové příspěvky | Zobrazit vaše příspěvky

Obsah fóra » Předměty (2010/2011) » 3BIT » Zimní semestr » IMS

[Souhrn témat]

nové téma | odpovědět **Stránka 1 z 2** [Příspěvků: 18] **Přejít na stránku 1, 2 Další**

Sledovat toto téma | Přidat do záložek | Verze pro tisk | Napsat e-mail | Ignorovat toto téma
Předchozí téma | Následující téma

Autor	Zpráva

skeWer

4BIT

Předmět příspěvku: [Souhrn témat] **Napsal(a):** 28 led 2011 08:59 pm

Prosím, nepostujte v tomhle tématu. Místo toho použijte **wishlist**.
Pokud najdete chybu, napište do wishlistu, opravím to. Pokud najdete nějakej hezkej příklad k některému z témat... je mi blbý psát potřetí co máte dělat 😁.

Témata (praktická)

- Kongruentní generátor pseudonáhodných čísel především by Royce
- Monte Carlo
- Transformace rozložení náhodných generátorů
  - Metoda vylučovací
  - Metoda inverzní transformace
- Snižování řádu derivace
  - Metoda postupné integrace by Royce
  - Metoda snižování řádu derivace
- Eulerova metoda (výpočet a implementace)
- Algoritmy řízení simulace
  - Next-event (diskrétní simulace)
  - Řízení spojité simulace - nikdo to nepotvrdil, ale asi by to mělo být ok
  - Řízení kombinované simulace
  - Řízení simulace - "zvláštní případy" (=ukončení simulace speciální událostí, stavové podmínky)
- Markovovské procesy - SHO typu M/M/x (pozor, masivní TL;DR... poprosil bych o příklady, protože naučit se to bez počítání jen podle návodu by byl epic feat...)
- Implementace (nejen) NAND hradla v SIMLIB/C++

Témata (teoretická)

- Optimalizace by Nindaleth
- Celulární automaty by Nindaleth
- Fuzzy logika (pokud tomu někdo rozumíte víc a najdete si čas, překontrolujte to po mě prosím)

Součástí každého odborníka v IT, který se zabývá sítěmi, je, že se nebojí číst RFC dokumenty a umí si je najít. -Matoušek Petr, Ing., Ph.D.
They are my space marines and they shall know no fear. -The Emperor of Mankind

Naposledy upravil skeWer dne 01 úno 2011 10:28 pm, celkově upraveno 43

Nahoru profil sz email upravit citace

skeWer

4BIT

Předmět příspěvku: Re: [Souhrn témat] **Napsal(a):** 28 led 2011 09:04 pm

TODO

(Priorita: černé > modré > zelené; červeně vyznačené už je rozdělané, ale potřebujou dodělat; šedě vyznačené by imo mělo být hotové)

- Metoda snižování řádu derivace

- Metoda postupné integrace - prosím o potvrzení, viz *
- Eulerova metoda
- Runge-Kutta
- Monte Carlo
- Markovovské procesy - snad už je kompletní
- Kongruentní generátory
- Metoda vylučovací
- Metoda inverzní transformace
- Metoda kompoziční
- Petriho sítě
- řízení simulace (pseudokódy) //částečně možná hotovo, před pokračováním chci potvrdit správnost řízení spojitých systémů
- Kalendáře
- SIMLIB

Teoretické otázky (defaultně asi všechny s modrou prioritou)

- Celulární automaty
- Optimalizace
- Fuzzy
- Z cílových znalostí:
  - Princip modelování a simulace, základní pojmy. Základní etapy modelování a simulace. Aplikační oblasti, výhody a nevýhody použití simulace
  - Systémy: Definice pojmu systém. Klasifikace systémů. Prvky systémů a jejich klasifikace. Popis struktury systémů a jejich chování. Časová množina. Ekvivalence chování systémů. Izomorfní a homomorfní systémy.
  - Modely a modelování: Klasifikace modelů, formy popisu modelů. Vytváření abstraktních modelů. Vytváření simulačních modelů.
  - Vztah systému a jeho modelů. Co je validace a verifikace modelů.
  - Simulace: Princip, různé typy simulace. Simulační nástroje, forma výsledků, analýza a vizualizace výsledků.
  - Použití Petriho sítí v simulaci. Typy Petriho sítí, chování Petriho sítí, typy přechodů.
  - Systémy hromadné obsluhy (SHO), obslužné sítě, fronty, priority, Kendallova klasifikace, modelování SHO.
  - Základy diskrétní simulace v SIMLIB/C++. Popis modelu a experimentu.
  - Orientačně: Co jsou tuhé (stiff) systémy, jaké problémy způsobují při simulaci.
  - Markovské procesy a řetězce.
  - Orientačně: Spolehlivost, intenzita poruch, jak se modeluje.
- Rychlé smyčky

_____

Naposledy upravil skeWer dne 01 úno 2011 05:39 pm, celkově upraveno 25

Nahoru profil sz email upravit citace

skeWer

4BIT

Předmět příspěvku: Re: [Souhrn témat]

Napsal(a): 28 led 2011 09:37 pm

Metoda vylučovací

K čemu to je ?
Transformace rozložení generátoru náhodných čísel.

Co potřebuju znát ?
Funkci hustoty daného rozložení.

Jak to funguje ?

1. Vezmeme funkci hustoty $f(x)$ - chceme jen ty části, co jsou nenulový. Zajímají nás tedy meze od-do na x-ové ose. Dál nás zajímají meze na y-ové ose - budou od nuly po nějakou maximální funkční hodnotu.
2. Meze na x-ové a y-ové ose dávají obdélník.
3. Náhodně střílíme do obdélníku: jednoduše vygenerujeme náhodně x a y v příslušných mezích.
4. Kontrolujeme, jestli jsme se trefili **pod** * danou křivku. Musí tedy platit: $y \leqslant f(x)$
5. Pokud předchozí nerovnost platí, **vracíme x**. Jinak střílíme znovu (goto 3).

Klady, zápory, omezení
+ Je to jednoduchý.
- Málo efektivní, když je plocha obdélníku podstatně větší než plocha pod $f(x)$.

* V opoře je <=. Nicméně možná pozor: někde uprostřed té funkce může být $f(x)=0$, a v tomto intervalu nechceme

generovat náhodná čísla - což by se v případě <= mohlo podařit. Pak je vhodné nějak ošetřit tyhle intervaly - střílenímm znovu, nebo třeba rozseknutím intervalů na x-ové ose na víc částí (pak se myslím jedná o komplozici).

TODO: linky na nějaké příklady

Credits:
Grim

_____
Součástí každého odborníka v IT, který se zabývá sítěmi, je, že se nebojí číst RFC dokumenty a umí si je najít. -Matoušek Petr, Ing., Ph.D.
They are my space marines and they shall know no fear. -The Emperor of Mankind

Naposledy upravil skeWer dne 29 led 2011 12:07 pm, celkově upraveno 6

Nahoru | profil | sz | email | upravit | citace

skeWer

Předmět příspěvku: Re: [Souhrn témat] **Napsal(a):** 28 led 2011 10:09 pm

4BIT

Metoda inverzní transformace

K čemu to je ?
Transformace rozložení generátoru náhodných čísel.

Co potřebuju znát ?
Inverzní distribuční funkci.

Jak získám inverzní distribuční funkci ?
Invertováním distribuční funkce. Ta se získá z funkce hustoty. Obvykle je zadaná jen funkce hustoty, takže postup:

1. Z funkce hustoty udělám distribuční funkci *. Je to distribuční funkce => neklesá, končí v jedničce
2. Invertuju ji - tzn. prohodím osy x a y (pro snažší představu je dobrý udělat si osu 45 stupňů a podle ní to zrcadlit). **Pozor**: je to funkce - pokud by se v ní měli po invertování vyskytovat svislé čáry, nekreslete je; plus viz **Martyho comment**.
   Všimněme si: po invertování se pohybujeme na x-ové ose v rozsahu 0-1.
3. Hotovo. Označím ji zde jako $id\left(x\right)$

Jak to funguje ?

1. Generujeme jediné číslo **x** v **rozsahu (0; 1)** **.
2. Vracíme funkční hodnotu: $id\left(x\right)$

Klady, zápory, omezení
+ Je to většinou efektivnější než vylučovací v tom, že generujeme vždy pouze 1 číslo.
- Ne vždy jsme schopni spočítat $id\left(x\right)$, a taky to saje pokud je $id\left(x\right)$ moc složitá na výpočet.

Příklad(y)

- by Milford

* získání distribuční fce $F\left(x\right)$ z fce hustoty $f\left(x\right)$ u metody inverzní transformace: viewtopic.php?p=251848#p251848, viewtopic.php?p=251860#p251860
** prosím o potvrzení intervalu. Ve slidech i opoře má (0;1), ale kongruentní generátory mají většinou ostrou hranici u nuly <0;1) . Pak je buď třeba ošetřit tu nulu (a případně generovat znova), nebo se na to vysrat a spoléhat, že to PPovi nebude vadit.

Credits:
Nindaleth (za doplnění *)

_____
Součástí každého odborníka v IT, který se zabývá sítěmi, je, že se nebojí číst RFC dokumenty a umí si je najít. -Matoušek Petr, Ing., Ph.D.
They are my space marines and they shall know no fear. -The Emperor of Mankind

Naposledy upravil skeWer dne 29 led 2011 02:28 pm, celkově upraveno 3

Nahoru | profil | sz | email | upravit | citace

skeWer

Předmět příspěvku: Re: [Souhrn témat] **Napsal(a):** 29 led 2011 12:42 am

4BIT

Metoda postupné integrace

K čemu to je ?
Převod rovnic vyšších řádů na rovnice 1.ho řádu (rovnice vyšších řádů většinou neumíme spočítat).

Co potřebuju ?

- kromě **rovnic**...
- **počáteční podmínky**
- více viz postup a poznámky

Jak to funguje ? *(možná budete chtít raději přeskočit rovnou k příkladu dole...)*

Je možné (ne nezbytné) používat Laplaceovy transformace: tedy nahrazení derivace za p: $y' = py$, $y'' = p^2 y$ *a integrace za 1/p:* $\int x = \frac{1}{p} x$ *. Potom se i do integrátoru píše* $\frac{1}{p}$ *místo* $\int$

V rovnici máme vstupy (x) a výstupy (všechno ostatní).

1. **Osamostatníme výstupní člen s nejvyšším řádem derivace** - tedy převedeme všechny ostatní členy (vstupní, výstupní s nižším řádem) na druhou stranu rovnice. Jakmile to máme, můžeme začít integrovat:
2. **Integrujeme rovnici.** Pokud používáme La Placeovu transformaci, dělíme péčkem.
   Tím snížíme řád derivace osamostatněného členu. Pravděpodobně dostaneme na druhé straně $vyraz_1 + \int vyraz_2$ (resp. $vyraz_1 + \frac{1}{p}\left(vyraz_2\right)$). V tom případě **provedeme substituci**: $w_1 = \frac{1}{p} vyraz_2$ a na pravé straně necháme $vyraz_1 + w_1$
3. Opakujeme druhý krok (=zavádíme pomocné proměnné) tak dlouho, dokud máme na levé straně derivaci (resp. p).
4. Rovnice máme hotové, teď **určíme počáteční podmínky**. To si zaslouží rozepsat:
   I. **Počáteční podmínky nepočítáme pro původní proměnné (x, y, y'...), ale pro ty zavedné substituční!** Proč ? Protože do integrátorů *obecně* nepůjde samostatná proměnná, nýbrž součet několika. Takže:
   II. Pro každou pomocnou proměnnou si najdeme v předešlém výpočtu rovnici, ze které jsme si ji vyjádřili. Tedy např: $p^2 y = vyraz_1 + w_1$. Snadno si vyjádříme $w_1 = p^2 y - vyraz_1$.
   III. A teď důležitej fakt: **v zadání máme specifikované počáteční podmínky pro původní proměnné**. Pro všechny, včetně těch derivovaných, ale kromě nejvyšších řádů. Takže pokud se nám v zadaných rovnicích vyskytují třeba: x, x', y, y', y''' pak známe počáteční podmínky x(0), x'(0), y(0), y'(0) i y''(0) (ano, i y''(0), přestože se nevyskytuje v zadání) *. A pokud nejsou podmínky explicitně určeny, předpokládáme, že jsou nulové**.
   IV. Tyhle počáteční podmínky dosadíme do rovnice v bodě II. Já jsem tam sice napsal obecnej $vyraz_1$, ale pod tím se ve skutečnosti bude schovávat nějakej polynom složenej právě z těch původních proměnných. Zapisuje se to takhle: $w_1(0) = p^2 y(0) - vyraz_1$... když si za výraz dosadím třeba y' + 2x' - x, tak: $w_1(0) = p^2 y(0) - py(0) - 2px(0) + x(0)$. Jak už bylo řečeno, za všechny proměnné napravo si dosadíme známé hodnoty počátečních podmínek.
   V. Po dosazení spočítáme, získali jsme počáteční podmínku pro tuhle pomocnou proměnnou (bude jí odpovídat jeden integrátor ve schematu). Opakujeme pro každou pomocnou proměnnou.

Jak z toho vytvořím blokové schéma ?
Asi nejlepší je kouknout se na příklad. Pár poznámek obecně:

- Kolik je nejvyšší řád derivace (v jedné rovnici), tolikrát budem integrovat. Takže dostanem tolik integrátorů (za každou rovnici).
- Do integrátoru většinou povede součet více proměnných = výstup sčíttačky.
- **Z integrátorů tečou zavedené pomocné proměnné (w1, w2....) !**
- Vždycky, když pro pomocnou proměnou kreslíme její odpovídající integrátor, koukneme se do substituce, kterou jsme provedli. Například máme: $w_1 = \frac{1}{p} vyraz_2$. Víme tedy, že výstupem integrátoru je $\frac{1}{p} vyraz_2$. A integrátor provádí integraci (dělení péčkem). Takže na jeho vstup zavedeme $vyraz_2$. Což bude většinou součet nějakých jiných proměnných.
  - Může se samozřejmě stát, že vyjde poslední řádek výpočtu např: $y = w_3$ - k tomu dojde, pokud nejvyšší řád y je větší, než řády ostatních proměnných. Pak nám z tohoto integrátoru teče y (a w3 současně, jsou ekvivalentní).
  - Pokud víc proměnných v rovnici má nejvyšší řád derivace, pak nám nakonec vyjde třeba něco takového: $y = vyraz + w_3$. V těchto případech jde výstup integrátoru do sčítačky, společně s nějakými původními proměnnými.

Příklad(y)

- Pdf odněkud z archivu: download/file.php?id=5161
- příklad by davidz

Poznámky, omezní

- Touhle metodou, narozdíl od snižování řádu derivace, umíme počítat rovnice s derivacemi vstupů. Pokud se tedy na písemce objeví taková rovnice a PP nespecifikuje metodu, nemáme na výběr a použijeme tuhle.
- *Teoretická poznámka: nejvyšší řád derivace vstupní proměnné nesmí přesáhnout nejvyšší řád výstupní proměnné (ie. nelze tím spočítat např.* $p^3 y = p^4 x$ *). Pochybuju ale, že by PP dal takovej chyták na semestrálku, byly by to moc koláčů bez práce.*

* Zde prosím o potvrzení... Myslím že to zmiňoval na přednáškách a asi by to mělo i vyplývat z logiky věci, ale nevšiml jsem si že by to psal v opoře nebo slidech.
**

> **opora str.60 píše:**
>
> Pokud počáteční podmínky nejsou uvedeny předpokládáme, že jsou nulové.

_____

Naposledy upravil skeWer dne 01 úno 2011 09:17 am, celkově upraveno 5

Nahoru profil sz email upravit citace

skeWer

4BIT

Předmět příspěvku: Re: [Souhrn témat] **Napsal(a):** 29 led 2011 11:48 am

Monte Carlo

Teorie

- Metoda založena na provádění náhodných experimentů.
- **Jedná se o třídu algoritmů, ne o jeden konkrétní!** (*zvýrazňuju to proto, že je potřeba umět ten princip zobecnit*)
- Užívá se v počítačová fyzice a příbuzných oborech. Také pro řešení integro-diferenciálních rovnic popisujících například osvětlení v 3D modelech.
- Přesnost $chyba = \frac{1}{\sqrt{N}}$ kde N = počet experimentů

Obecný princip (zpracováno z en Wikipedie)

1. Definujeme množinu přípustných vstupů (pro každou vstupní proměnnou).
2. Provedeme experiment:
   I. Náhodně generujeme vstupy. Nám bude stačit uniformní rozložení, jinak se můžou používat i jiné.
   II. Vstupy použijeme v deterministickém výpočtu.
3. Agregujeme výsledky z N experimentů.

Příklad(y)

- opora st. 27 a 28 (*nepleťe si ten modrej graf s vylučovací metodou transformace rozložení; zapamtujte si, že jednorozměrné integrály zásadně nepočítáme metodou Monte Carlo, na to máme přesnější a efektivnější metody - PP*)
- Kód pro výpočet integrálu a objemu tělesa: **http://eva.fit.vutbr.cz/~xkalab00/ims:simlib#monte_carlo**

Zobecnění příkladu

Když si vezmeme výše zmíněný obecný princip a srovnáme s příkladem na počítání objemu tělesa, tak:

1. rozsah `xmin[0] + (xmax[0] - xmin[0])` ie. od xmin[0] do xmax[0] ... to je samozřejmě pouze pro jednu proměnnou
2. experiment
   I. `x = xmin[0] + (xmax[0] - xmin[0]) * Random();`
   "střílíme" náhodně do definovaného prostoru určeného vstupními proměnnými.
   II. `if (BodJeUvnitrTelesa(x, y, z)) jeUvnitr++;`
   Deterministický výpočet je zde ta funkce BodJeUvnitrTelesa(). Čili musíme znát tvar tělesa a pro každý bod jsme schopni určit, jestli je uvnitř nebo ne.
   Proměnná `jeUvnitr` pak slouží k agregaci výsledků experimentů (ie. uchováváme v ní, v kolika experimentech jsme se trefili do tělesa).
3. agregace:

Kód: Číslování řádků on/off | Rozbalit/Sbalit | Vybrat vše

```
1. for (int i = 0; i < POCET_EXPERIMENTU; i++) {
```

```
2.    //experiment
3. }
4. double P = jeUvnitr / POCET_EXPERIMENTU; //see ? předpokládáme, že P je zlomek určující, jak velkou část rozsahu zabírá naše těleso.
5. double rozsah = (xmax[0] - xmin[0]) * (xmax[1] - xmin[1]) * (xmax[2] - xmin[2]); //Kde rozsah je prostor, ve kterém jsme generovali náhodné body (bod 2.I)
6. return P * rozsah;
```

Credits
legien (za opravení drobné vzorečku chyby)

Naposledy upravil skeWer dne 01 úno 2011 06:47 pm, celkově upraveno 1

Nahoru profil sz email upravit citace

skeWer

4BIT

Předmět příspěvku: Re: [Souhrn témat] **Napsal(a):** 29 led 2011 12:27 pm

Next-event (diskrétní simulace)

Jak to funguje ?

- Během simulace se do kalendáře událostí přidávají **záznamy** (*pozor, slovíčkaření* ... *resp. preciznost vyjadřování*).
- Simulace běží tak dlouho, dokud máme v kalendáří záznamy, **nebo dokud nenarazíme na záznam mimo simulační čas.**
- Na konci simulace se **posuneme na konec simulace** (... ehm).

Pseudokód

Kód: Číslování řádků on/off | Rozbalit/Sbalit | Vybrat vše

```
1. Inicializace_kalendare
2. while(kalendar_neprazdny) {
3.    vyjmi_prvni_zaznam_z_kalendare
4.    if (aktivacni_doba_zaznamu > doba_konce_simulace) break;
5.    cas = aktivacni_doba_zaznamu;  //hlavně nepřehoďte tyhle dva řádky
6.    proved_udalost  //jinak...
7. }
8. cas = doba_konce_simulace
```

Naposledy upravil skeWer dne 01 úno 2011 08:15 pm, celkově upraveno 3

Nahoru profil sz email upravit citace

skeWer

4BIT

Předmět příspěvku: Re: [Souhrn témat] **Napsal(a):** 29 led 2011 01:41 pm

Kongruentní generátor

K čemu to je ?
Generování pseudonáhodných čísel s rovnoměrným rozložením

Co potřebuji znát?

- vhodné konstanty a, b, M
  testované (vhodné) konstanty jsou: (viz opora str. 19)
  a - 69069
  b - 1
  M - 2^32

Jak to funguje ?
Kongruentní generátor generuje celá čísla v rozsahu 0 ≤ xi < m s rovnoměrným rozložením. Protože často potřebujeme tzv. normalizované rozložení s rozsahem <0, 1) musíme použít operaci dělení modulem m.

Příklad(y)
jednoduchý generátor v C, generuje <0; 1)

Kód: Číslování řádků on/off | Rozbalit/Sbalit | Vybrat vše

```
1.  static unsigned long ix = seed;
2.
3.  double Random() {
4.      ix = (ix * a + b) % M;
5.      return ix / M;
6.  }
7.
```

tenhle generuje <0; 1>

Kód: Číslování řádků on/off | Rozbalit/Sbalit | Vybrat vše

```
1.  static unsigned long ix = seed;
2.
3.  double Random() {
4.      ix = (ix * a + b) % M;
5.      return ix / (M-1);
6.  }
7.
```

Pozn: ve skriptech lze najít něco takového:

Kód: Číslování řádků on/off | Rozbalit/Sbalit | Vybrat vše

```
1.  static unsigned long ix = seed;
2.
3.  double Random() {
4.      ix = ix * a + b; //není přítomno modulo
5.      return ix / (ULONG_MAX +  1);
6.  }
```

Modulo je dáno implicitně - přetečením. Generují se nám čísla v rozsahu 0 - ULONG_MAX. Tím, že výsledek podělíme (ULONG_MAX + 1), dostaneme čísla v rozsahu <0;1) . Pokud bychom dělili pouze (ULONG_MAX), máme rozsah <0; 1>.

Klady, zápory, omezení
+ jednoduchý generátor
- při špatně zvolených parametrech generuje závislou posloupnost čísel (nebo tak nějak)

Credits
Royce (za většinu postu)
Milford (za opravy)

Součástí každého odborníka v IT, který se zabývá sítěmi, je, že se nebojí číst RFC dokumenty a umí si je najít. -Matoušek Petr, Ing., Ph.D.
They are my space marines and they shall know no fear. -The Emperor of Mankind

Nahoru | profil | sz | email | upravit | citace

skeWer

4BIT

Předmět příspěvku: Re: [Souhrn témat]

Napsal(a): 29 led 2011 03:00 pm

Metoda snižování řádu derivace

K čemu to je ?
Převod rovnic vyšších řádů na soustavu rovnic 1. řádu, pro které máme vhodné numerické metody

Co potřebuji znát?
Počáteční podmínky

Podmínka
V rovnicích nesmí být derivace vstupů!

Jak to funguje
1. osamostatníme nejvyšší řád derivace
2. sestavím sekvenci integrací

Obecně: $y \times p^{n-1} = \int y \times p^{n}$ (kde $y \times p^{n}$ je n-tá derivace y).

Příklad

- $y''-2y'+y=x$
  $y''=2y'-y+x$
  $y'=\int y''$
  $y=\int y'$

- pdf, výpočet + sestavení blokového schematu

Sestavení blokového schématu
podobně jako u metody postupné integrace (odkaz). Nicméně zde pracujeme rovnou s původními proměnnými.

Credits
Royce (celej post)

_____
Součástí každého odborníka v IT, který se zabývá sítěmi, je, že se neboji číst RFC dokumenty a umí si je najít. -Matoušek Petr, Ing., Ph.D.
They are my space marines and they shall know no fear. -The Emperor of Mankind

Naposledy upravil skeWer dne 01 úno 2011 09:26 am, celkově upraveno 1

Nahoru | profil | sz | email | upravit | citace

skeWer

Předmět příspěvku: Re: [Souhrn témat]

Napsal(a): 29 led 2011 04:48 pm

4BIT

Řízení spojité simulace

Jak to funguje ?

- Spojitá simulace jen sleduje průběh spojitých veličin.
- Používá se na to nějaká numerická metoda (Euler, Runge-Kutta).
- Zvolí se integrační krok
- Pak krokujeme: v každém kroku posuneme čas a spočítáme hodnotu sledované veličiny (někam si ji poznačíme, ačkoliv v písemce to asi možná není nutné).
- Dokročení: je třeba sledovat, abychom nepřekročili simlační čas - pokud by k tomu mělo dojít (v posledním kroku) , snížíme krok.

Pseudokód
Převzato odsud.

Kód: Číslování řádků on/off | Rozbalit/Sbalit | Vybrat vše

```
1. inicializace poc. stavu a promennych a casu;
2.
3. while (cas < koncovy_cas) {
4.   Print_results();
5.   Update_integrators();
6.   if ((cas + krok) > koncovy_cas) {  krok = koncovy_cas - cas;  }
7.   euler();
8.   cas += krok;
9. }
```

Pozn: **Update_integrators()** si uloží pro všechny integrátory aktuální vstup. Je důležité volat to před výpočtem kroku (zde euler) ! Pokud jde totiž výstup jednoho integrátoru na vstup jiného, tak kdybychom vypočítali výstup prvního, druhej by dostal na vstup hodnoty z příštího kroku, což je bullshit.

Credits
Royce

_____
Součástí každého odborníka v IT, který se zabývá sítěmi, je, že se neboji číst RFC dokumenty a umí si je najít. -Matoušek Petr, Ing., Ph.D.
They are my space marines and they shall know no fear. -The Emperor of Mankind

Naposledy upravil skeWer dne 01 úno 2011 **06:55 pm, celkověupraveno3**

Nahoru | profil | sz | email | upravit | citace

skeWer

Předmět příspěvku: Re: [Souhrn témat]

Napsal(a): 29 led 2011 07:47 pm

4BIT

Markovovské procesy - SHO typu M/M/x

Co potřebujeme ?

- Nějaký SHO typu M/M/x.

- Intenzitu příchodů transakcí (**počet příchodů za jednotku času**)
- Intenzitu obsluhy (**počet obsloužených transakcí za jednotku času**)
- Znát maximální délku fronty

Co po nás může chtít ?

- Nakreslit schema i s popisky
- Určit mez stability
- Vyjádřit rovnice pro ustálené stavy
- Vypočítat je
- Určit pravděpodobnost některých stavů (s různými obměnami, třeba pravděpodobnost že je plná fronta atp.)
- Obecný tvar analytického řešení - **link s vysvětlením**. (**doporučuju přečíst, není to složité, ale nedá se to moc vymyslet z hlavy a PP se na to ptá**)
- Určit průměrnou délku fronty (*nechtěl na prvním a myslím ani na 1.opravném termínu, ale kdo ví...*)
- Určit průměrnou dobu čekání ve frontě (*ditto*)
- **TODO**: určitě jsem na něco zapomněl, doplňte prosím

Příklady

- M/M/4 zadání + potvrzené výsledky
- M/M/2 bez fronty zadání včetně výsledků
- **TODO** - doplnit další

Postup

1. **Tvorba schematu a základní pojmy**

   Potřebujeme znát:

   - Maximální délku fronty - máme zadanou. Pokud je nekonečná, některé výpočty jsou trochu jiné, ale k tomu pozdějc.
   - Intenzitu příchodů = počet transakcí za jednotku času. Značíme λ.
   - Intentzitu obsluhy = počet obsloužených zařízení za jednotku času. Značíme μ.

   Pozor! λ i μ musí být vyjádřené stejným způsobem. Pokud víme, že do systému přijde např 5 transakcí za minutu a doba obsluhy je 20 sekund, známe $\lambda = 5$ a musíme převést $\mu = \left(\frac{60\,s}{20\,s} \right) = 3$

   Ok. Dál bereme, že náš systém se nachází v nějakém stavu - řešíme to jen do té míry, že chceme vědět, kolik je v systému transakcí. Takže si značíme stavy systému p0, p1, p2, ... Index značí, kolik je v systému transakcí.
   Náš systém se chová tak, že když přijde transakce, zabere zařízení. Když není žádné volné, jde si stoupnout do fronty. Když už není místo ani ve frontě, odchází a nezajímá nás.

   V následujícím textu budu používat:

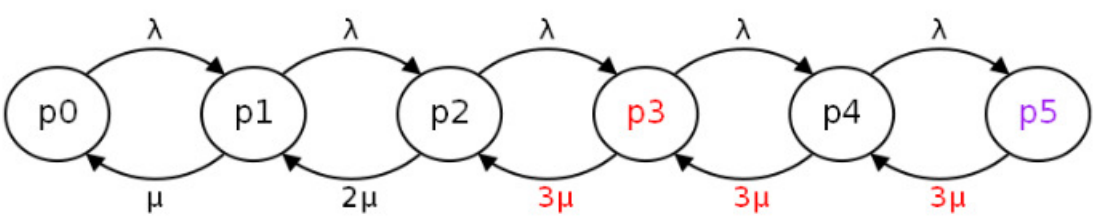
   - **x** = počet zařízení
   - **f** = délka fronty

   Kolik může být v systému maximálně transakcí ?
   1 transakce na každé zařízení plus maximální délka fronty. Počet stavů systému je o 1 vyšší: máme ještě stav p0, kdy je systém prázdný. Tedy počet stavů $N = x + f + 1$.

   Potřebujeme ještě označit přechody mezi stavy. Intentzita příchodů transakcí určuje, jak často budeme přecházet do následujícího (vyššího) stavu - takže tyhle přechody označíme lambdou. Počet obsloužených transakcí za jednotku času nám udává, jak často odejde transakce, tedy zpětné přechody značíme mí. Zde ale pozor: když je v systému více zařízení a současně více transakcí, tak transakce odcházejí rychleji - tolikrát rychleji, kolik zařízení pracuje v daném stavu. Pracuje tolik zařízení, kolik je transakcí v systému, ale nejvýše x. Takže přechody značíme takhle.

   M/M/3

   x = 3 f = 2

   A jsme hotoví.

2. Stabilita a mez stability

Systém je stabilní, když stíhá obsluhovat příchozí požadavky. To nastává v případě, že $\lambda \leqslant x \times \mu$ (ale viz *).
Na mezi stability je v případě, že $\lambda = x \times \mu$.

V našem M/M/3 příkladu je systém na mezi stability pro např $\lambda = 3, \mu = 1$. Stejně tak $\lambda = 15, \mu = 5$.

3. Rovnice pro ustálené stavy

> **slajdy píše:**
>
> V ustáleném stavu se pravděpodobnosti již nemění. Proto intenzita přechodů násobená pravděpodobností stavu musí být v rovnováze.

Chceme napsat rovnici pro každý stav systému. Předpokládáme, že stavy jsou ustálené, to znamená, že "všechno co jde ze stavu se rovná tomu, co jde do něj". $IN = OUT$, zapisujeme $IN\text{-}OUT = 0$.

Pro každý stav:

1. Vezmeme všechny ohodnocení hran **vedoucích do a z aktuálního stavu**.
2. Ohodnocení vynásobíme pravděpodobností stavu **ze kterého vedou**. Pokud vychází z aktuálního stavu, násobíme je -1 (OUT hrany).
3. Tyto členy sečteme.
4. Postavíme rovno nule.

Pravděpodobnost si značíme např p_0, p_1 atd. Takže pro náš příklad M/M/3 s frontou 2:

`stav` : *rovnice*
`p0`: $\mu p_1 - \lambda p_0 = 0$
`p1`: $\lambda p_0 + 2\mu p_2 - \mu p_1 - \lambda p_1 = 0$
`p2`: $\lambda p_1 + 3\mu p_3 - \mu p_2 - \lambda p_2 = 0$
`p3`: $\lambda p_2 + 3\mu p_4 - \mu p_3 - \lambda p_3 = 0$
`p4`: $\lambda p_3 + 3\mu p_5 - \mu p_4 - \lambda p_4 = 0$
`p5`: $\lambda p_4 - 3\mu p_5 = 0$

Tak, to jsou rovnice pro ustálené stavy.

Pokud bychom chtěli vypočítat pravděpodobnosti jednotlivých stavů, musíme si vyjádřit všechny pravděpodobnosti pomocí jedné (třeba p0). A pak dosadit do rovnice:
$p_0 + p_1 + p_2 + p_3 + p_4 + p_5 + p_6 = 1$

Nebudu to konkrétně rozvádět, ale mělo by to vyjít:

$$p_0 = p_0$$
$$p_1 = p_0 \times \left(\frac{\lambda}{\mu}\right)$$
$$p_2 = p_0 \times \frac{1}{2}\left(\frac{\lambda}{\mu}\right)^2$$
$$p_3 = p_0 \times \frac{1}{2\times 3} \times \left(\frac{\lambda}{\mu}\right)^3$$
$$p_4 = p_0 \times \frac{1}{2\times 3\times 3} \times \left(\frac{\lambda}{\mu}\right)^4$$
$$p_5 = p_0 \times \frac{1}{2\times 3\times 3\times 3} \times \left(\frac{\lambda}{\mu}\right)^5$$

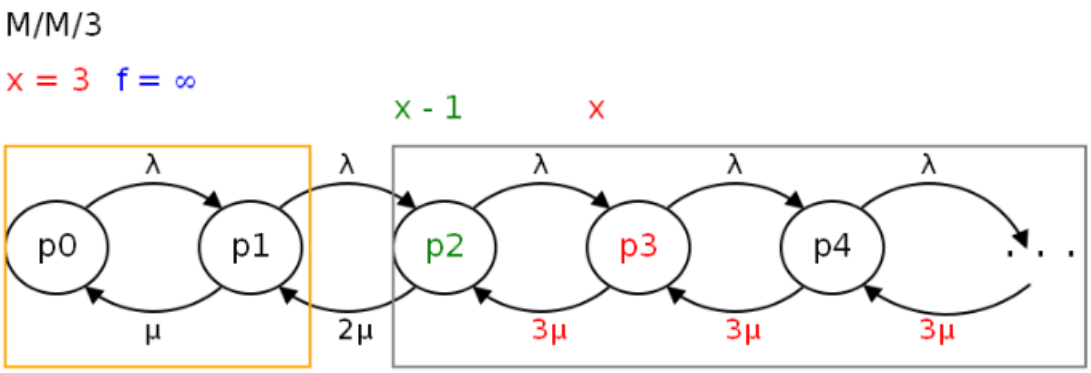
Ve jmenovatelích jsou koeficienty hrany z aktuálního stavu do předchozího (tedy μ hrana směřující doleva)

4. M/M/x s nekonečnou frontou

Tyhle systémy mají nekonečno stavů. Když chceme počítat pravděpodobnost, prvních pár spočítáme stejně jako normálně, a pro zbytek se musí použít vzorec na výpočet nekonečné geometrické řady.

Opakování- NGŘ: $a_0 + a_0 \times q^1 + a_0 \times q^2 \ldots$

Vzorec pro součet NGŘ: $\sum = \frac{a_0}{1-q}$ kde q je kvocient NGŘ.



Prvních pár stavů si vyjádříme rovnicemi pro ustálený stav, jako normálně.
Následující stavy vypočítáme jako součet NGŘ.
Stav x-1 je počáteční prvek NGŘ (a0 ve vzorci).

Musíme si vyjádřit kvocient q. $q = \frac{a_1}{a_0}$ (*lze využít libovolné dva prvky, ale ať se nadřeme co nejmíň*).

HINT: Kvocient se vždy rovná $\frac{1}{x} \times \frac{\lambda}{\mu}$. Schválně se podívejte na rovnice ustálených stavů a vezměte si počáteční člen NGŘ, tedy p2, a srovnejte ho s následujícími členy - jmenovatel roste vždycky x-krát a mocnina $\left(\frac{\lambda}{\mu}\right)$ se zvětší o jedničku.**

Pak si opět vyjádříme vzoreček, v našem případě:

$$p_0 + p_1 + \frac{p_2}{1-q} = 1$$

což se rovná:

$$p_0 + p_0 \times \left(\frac{\lambda}{\mu}\right) + \frac{p_0 \times \frac{1}{2}\left(\frac{\lambda}{\mu}\right)^2}{1 - \frac{1}{3} \times \frac{\lambda}{\mu}} = 1$$

Jak vidno, jediná neznámá je p0, takže to umíme spočítat (*neudělat chybu ve výpočtu je nicméně kurevsky těžší* 😠)
.

5. Průměrná délka fronty, průměrná doba čekání ve frontě

Počítá se to jako vážený průměr délek front v jednotlivých stavech (vahou je pravděpodobnost daného stavu).
Tedy:

$$avgQlen = \sum_{x \in stavy} p_x \times delkaFronty\left(p_x\right)$$

S tím, že $delkaFronty\left(p_x\right)$ záleží na tom, kolik je v systému zařízení (pro M/M/3 je nulová do stavu P3, pak roste po jedničkách).

A průměrná délka čekání by měla být průměrná délka fronty krát doba obsluhy (ie počet procesů než se dostaneš na řadu krát doba obsluhy každého z nich)

$$avgQlen \times \frac{1}{\mu}$$

S nekonečně dlouhou frontou si ale nejsem jistej, jestli to jde spočítat.... snad to nebude chtít.

Credits
Nindaleth (za doplnění linku pro analytické řešení apod)

* Nejsem si jistej, jestli tam nemá být pouze lambda < x*mí...
**Tady je taky vidět, proč je systém stabilní pro $\lambda < x \times \mu$. Pokud by lambda byla větší, kvocient NGŘ by byl > 1 a NGŘ by divergovala.

_____
Součástí každého odborníka v IT, který se zabývá sítěmi, je, že se nebojí číst RFC dokumenty a umí si je najít. -Matoušek Petr, Ing., Ph.D.
They are my space marines and they shall know no fear. -The Emperor of Mankind

Naposledy upravil skeWer dne 01 úno 2011 11:01 am, celkově upraveno 5

Nahoru profil sz email upravit citace

skeWer

4BIT

Předmět příspěvku: Re: [Souhrn témat] **Napsal(a):** 30 led 2011 08:22 pm

Eulerova metoda

K čemu to je ?
Pro výpočet diferenciálních rovnic prvního řádu (hodnotu proměnné v následujícím stavu)

Co potřebuju znát ?
- Délku kroku (je daný)
- Hodnotu a derivaci proměnné v aktuálním stavu (čase)
- S předešlým bodem souvisí - co potřebujeme pro počítání diferenciálních rovnic obecně ? **Počáteční podmínky**.
- Pamatovat si vzorec

Jak to funguje ?
V bodech:
- Eulerova metoda je jednokroková - počítá následující stav proměnné pouze na základě aktuálního
- Vzorec Eulerovy metody:

$$y(t+h) = y(t) + h \times f(t, y(t))$$ kde:

  - h je délka kroku
  - $f(t, y(t))$ je derivace y v aktuálním čase, tzn. $y'(t)$. *V písemce ale používejte f(t, y(t)), PP to má raději.*

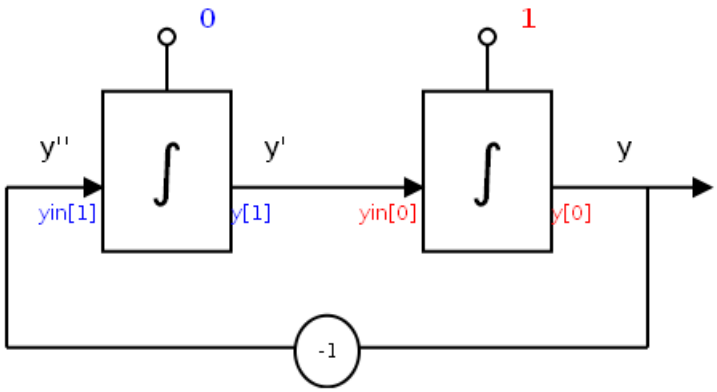
Příklady
- pdf, od slidu 6 je výpočet
- TODO - doplnit další (je blbý, že tu na ně není samostatný téma... 😐)

Implementace v C

Kód: Číslování řádků on/off | Rozbalit/Sbalit | Vybrat vše

```
1.  double yin[2], y[2] = { 1.0, 0.0 };
2.  double time = 0, h = 0.001;
3.
4.  void update() { // výpočet vstupů integrátorů
5.     yin[0] = y[1]; // y'
6.     yin[1] = -y[0]; // y''
7.  }
8.
9.  void integrate_euler() { // krok integrace
10.    update();
11.    for (int i = 0; i < 2; i++)
12.       y[i] += h * yin[i];
13.    time += h;
14. }
15. int main() {
16.    while (time < 20) {
17.    printf("%10f %10f\n", time, y[0]);
18.    integrate_euler();
19. }
20. return 0;
```

Tohle je pro rovnici
$y'' = -y$



Checkněte Řízení spojité simulace - funkce **update** odpovídá **Update_Integrators()**. Zopakuji: je nutné před samotným výpočtem updateovat vstupy. Kdyby se vstupy updateovaly v tom cyklu společně s počítáním výstupů, bylo by to blbě.

Credits
legien (oprava vzorce)

_____
Součástí každého odborníka v IT, který se zabývá sítěmi, je, že se nebojí číst RFC dokumenty a umí si je najít. -Matoušek Petr, Ing., Ph.D.
They are my space marines and they shall know no fear. -The Emperor of Mankind

Nahoru | profil | sz | email | upravit | citace

skeWer

Předmět příspěvku: Re: [Souhrn témat] **Napsal(a):** 31 led 2011 03:46 pm

4BIT

Implementace (nejen) NAND hradla v SIMLIB/C++

Popisuje to Hrubý na druhém democviku, poslední 4 minuty.
Koukněte na kód na pitlově wiki.

Hradlo
- Obsahuje nějaké vstupy: IN[2]
- Obsahuje výstup: OUT
- Výstup počítá ze vstupů: (část v Behavior za Passivate)
- Výstup počítá se zpožděním:
  - V cyklu v Behavior se vždycky pasivuje.
  - Změna vstupu je reprezentována tím, že se zavolá fce input(int idx, int val) - kde idx identifikuje, který ze vstupů se změnil a val jeho novou hodnotu. Pak input probudí hradlo v (Time + Delay).
  - Hradlo se probudí v Behavior za Passivate(), provede výpočet, změní hodnotu OUT a cyklí dál. Celý.
- Na počátku má na vstupech i na výstupu nějakou nedefinovanou hodnotu VAL_X (asi vysoká impedance).

_____
Součástí každého odborníka v IT, který se zabývá sítěmi, je, že se nebojí číst RFC dokumenty a umí si je najít. -Matoušek Petr, Ing., Ph.D.
They are my space marines and they shall know no fear. -The Emperor of Mankind

Nahoru | profil | sz | email | upravit | citace

skeWer

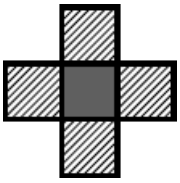
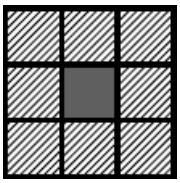
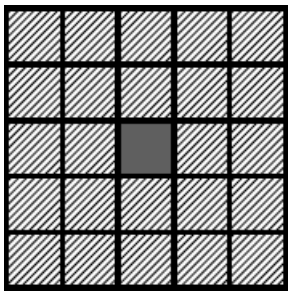
Předmět příspěvku: Re: [Souhrn témat] **Napsal(a):** 31 led 2011 08:47 pm

4BIT

Celulární automaty

Teorie:
- CA tvoří diskrétní systém
- Použití: simulace dopravy, šíření epidemií, modely evoluce, ...

Pojmy:
- **Buňka** - základní element, může nabývat jeden ze stavů z konečné množiny (např. 0 nebo 1)
- **Pole buněk**
  - je rovnoměrné (mřížka)
  - n-rozměrné, obvykle 1D nebo 2D
  - může být i nekonečné
- **Okolí** - několik typů okolí - liší se počtem okolních buněk, se kterými se pracuje (šrafované buňky tvoří okolí centrální plnobarevné buňky)

- von Neumanovo 
- Moorovo 
- rozšířené Moorovo 

- **Pravidla** - definují, jak bude buňka vypadat v příštím kroku na základě stavu této buňky a jejího okolí
  - Počet možných vstupů a pravidel: např. pro Moorovo okolí (8 okolních + 1 centrální buňka) je počet možných vstupů $2^{\left(8+1\right)}=512$ a 2^{512} počet všech možných funkcí/pravidel
  - Obecně, pro buňky s n stavy tedy: počet kombinací vstupů $PK=n^{\left(pocet\text{-}bunek\text{-}v\text{-}okoli+1\right)}$ a počet různých pravidel pro dané okolí je $PP=n^{PK}$
  - (*pokud si to chcete odvodit, doporučuji jít na přednášku, resp. záznam*)

Vlastnosti CA:
- konfigurace je definována jako stav všech buněk
- stav CA se vyvíjí v čase a prostoru dle zadaných pravidel
- čas i prostor jsou v CA diskretizovány
- počet stavů buňky je konečný
- buňky jsou identické
- následující stav buňky je závislý jen na aktuálním stavu

Klasifikace CA:
- třída 1: Po konečném počtu kroků dosáhnou jednoho konkrétního ustáleného stavu
- třída 2: Dosáhnou periodického opakování (s krátkou periodou) nebo zůstanou stabilní
- třída 3: Chaotické chování (tvoří speciální fraktální útvary)
- třída 4: Kombinace běžného a chaotického chování (například Life), nejsou reverzibilní

Reverzibilní automaty - neztrácí informaci při vývoji v čase - lze obrátit směr toku času a vracet se k předchozím stavům

Co tu není: Pravděpodobnostní CA

Naposledy upravil skeWer dne 01 úno 2011 07:02 pm, celkově upraveno 1

Nahoru profil sz email upravit citace

skeWer

4BIT

Předmět příspěvku: Re: [Souhrn témat] **Napsal(a):** 31 led 2011 08:47 pm

Optimalizace

nalezení optimálních hodnot parametrů modelu

Matematická definice optimalizace:

Hledáme minimum nebo maximum cenové funkce $F\left(\vec{x}\right)$.

Minimalizace je algoritmus, který počítá: $\vec{x}:F\left(\vec{x}\right)=\min\wedge C\left(\vec{x}\right)$

kde:

- $\vec{x}$ je vektor hodnot parametrů
- F je cenová funkce
- C reprezentuje různá omezení (constraints) pro $\vec{x}$

Maximalizace $\Longrightarrow$ použijeme $-F$.
Problém: lokální minima $\Longrightarrow$ používáme globální optimalizační metody.

K čemu jsou constraints:

Reprezentují různá omezení $\vec{x}$ - např. při optimalizaci parametrů vařeného pokrmu nemůžeme u přidaného koření použít zápornou hmotnost ani ho tam dát dvě kila.

lineární programování - matematická disciplína optimalizace parametrů - řeší problém nalezení minima (resp. maxima) lineární funkce n proměnných na množině, popsané soustavou lineárních nerovností; není to programování

Optimalizační metody:

- gradientní
- simulované žíhání - začíná se s velkými změnami, které se postupně pomalu zmenšují; lze přijmout i horší řešení - pokus vyhnout se uvíznutí v lokálním minimu
- genetické - přejímá metody z evoluce - selekce, párování, křížení, mutace (více viz link)

Postup při optimalizaci parametrů:

Kód: Číslování řádků on/off | Rozbalit/Sbalit | Vybrat vše

```
1. while (hodnota cenové funkce > uspokojivá hodnota) { //pro minimalizaci
2.   iterace optimalizační metody
3.   znovuprovedení experimentu
4.   ohodnocení cenovou funkcí
5. }
```

Součástí každého odborníka v IT, který se zabývá sítěmi, je, že se nebojí číst RFC dokumenty a umí si je najít. -Matoušek Petr, Ing., Ph.D.
They are my space marines and they shall know no fear. -The Emperor of Mankind

Nahoru | profil | sz | email | upravit | citace

skeWer

4BIT

Předmět příspěvku: Re: [Souhrn témat] **Napsal(a):** 01 úno 2011 05:18 pm

Fuzzy logika

Co to je ?
Jeden ze způsobů popisu neurčitosti (lingvistická neurčitost). (*jiná forma neurčitosti je třeba pravděpodobnost*)

Kde se využívá ?
Regulace a automatizace, např

- řízení jeřábů, pračky, expozice u foťáků, výtahů
- v metru — zvýšená přesnost zastavování, plynulejší brždění, nižší spotřeba energie
- ABS, řízení motoru, volnoběhu a klimatizace
- Korekce chyb ve slévárenství

Rozpoznávání, analýza

- Rozpoznávání ručně psaných textů, řeči
- při hledání identifikačních a profilových systémů pachatele (velký, ne příliš těžký, víceméně starý, …)
- Analýza portfolia při investování na kapitálovém trhu

Jak to funguje ?
V bodech

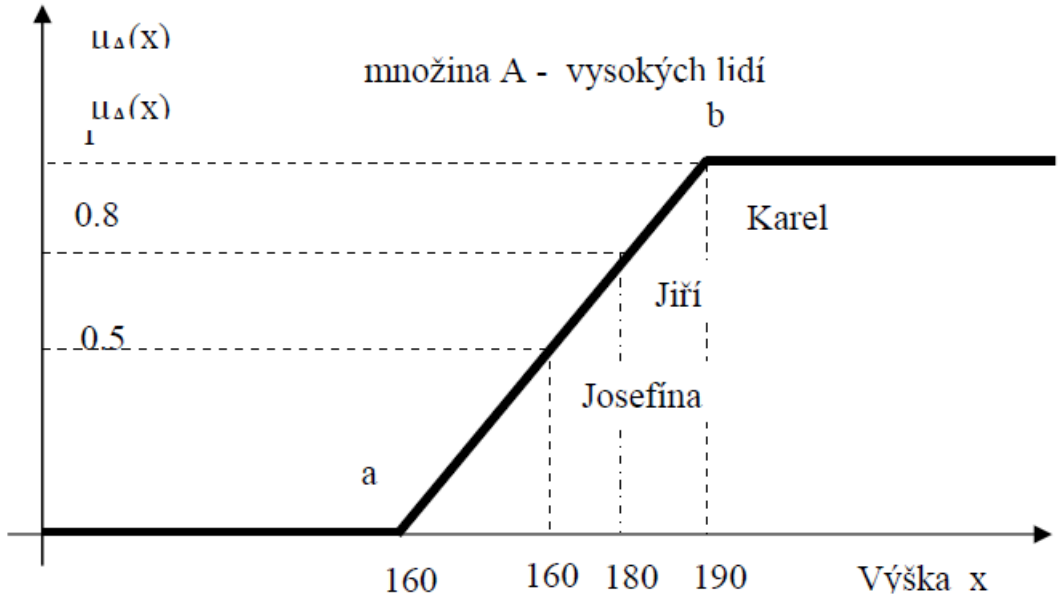
- Na vstup fuzzy modelu přijde "ostrá" hodnota (třeba nějaké spojité proměnné).
- Hodnota je fuzzifikována.
- Inference - fuzzifikovaná hodnota je prohnána fuzzy pravidly.
- Výsledky pravidel jsou agregovány do nějaké výstupní fuzzy množiny.
- Agregát je deffuzifikován na "ostrou" hodnotu výstupní proměnné.

Podrobněji

Je to rozšíření boolovské logiky.
Ve fuzzy se setkáváme s pojmem **fuzzy množina**. Pro obyčejnou množinu platí, že každý prvek do ní buď platí, nebo ne, ie {0; 1}. U fuzzy množin ale platí, že každý prvek patří do každé množiny s určitou mírou, ie <0; 1>*. Tomu se říká **míra příslušnosti**.

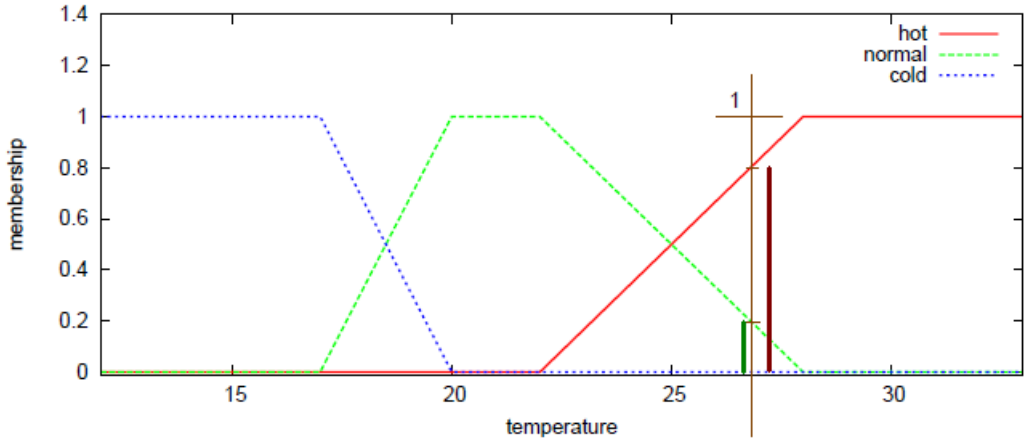
Takže fuzzy množinou můžou být třeba *vysocí lidé*. Když do fuzzy systému přijde na vstup nějaká proměnná - *člověk* -, chceme vyhodnotit, jak moc ta proměnná přísluší ke které množině - chceme zjistit, jak moc tenhle člověk patří mezi *vysoké lidi*.

K tomu se užívají **funkce příslušnosti**. Funkce dostane něco na vstup a vypočítá, jak moc danej vstup přísluší které množině. Takže nějaká naše funkce by vzala toho člověka a podle výšky by určila, jak moc je vyhovuje pojmu "vysoký člověk". Ty funkce často nejsou nic ďábelskýho, viz slidy, opora IMP, wikipedia.



(přejato z IMP opory)
Tohle je přesně ono - funkce vyjadřující, jak moc každej člověk patří do množiny vysokých lidí.

Takových množin může být pochopitelně více (vysocí lidé, nízcí lidé, normálně vysocí lidé...). Pro každou množinu se vytvoří vlastní funkce. Pro každou vstupní hodnotu musí platit, že součet měr příslušnosti pro všechny existující fuzzy množiny je roven jedné. Naznačeno na obrázku:



Zde je vstupem teplota. Odhadem: ve vyznačeném místě (t=26.5) je míra příslušnosti k "horký" 0.8, k "normální" 0.2, a ke "studený" 0.0.

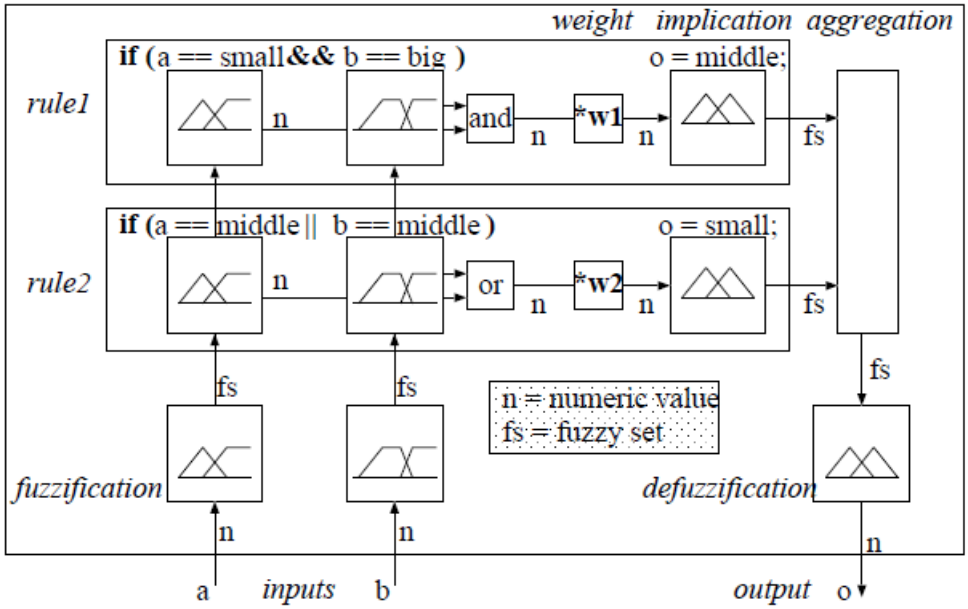
Inferenční pravidla
Inferenční pravidla slouží k nějakému řízení činnosti, jsou ve tvaru *IF <podmínka> THEN <implikace>*.
Důležité je, že se počítá s tím, že <podmínka> neplatí stoprocentně (!= 1). Proto ani vyhodnocení pravidla neplatí stoprocentně. Podmínka je právě řešena pomocí těch měr příslušnosti - a podle toho, jak moc je podmínka splněna, tak i výstup je splněn stejnou měrou. Pokud máme třeba pravidlo: *IF (teplota == studená) THEN zapni topení na 20°C* a na vstupu naměříme teplotu, která do množiny studených teplot náleží s příslušností 35%, výsledkem tohoto pravidla vlastně je "zapni topení na 7°C" (= 0.35 * 20).

Opět, pravidlo nebývá jenom jedno. Musí se vyhodnotit všechny a jejich výsledky se nějak **agregovat**. V našem případě by třeba bylo další pravidlo *IF (teplota == normální) THEN zapni topení na 2°C*. Naše naměřená teplota by splnila podmínku na 65%, výsledkem by bylo 0.65*2 = 1.3°C. Výsledky obou pravidel by po agregaci daly "zapni topení na 8.3°C". (ve skutečnosti je to asi trochu složitější než prosté sečtení)

Pravidla mohou být i složitější, pak se používají různé operace nad množinami - konjunkce, disjunkce, negace atp. Narozdíl od boolovské logiky nemusejí mít jednoznačný význam - na přednášce uváděl, že třeba u konjunkce se používá $A \wedge B = minimum(A,B)$ a jinde třeba $A \wedge B = A \times B$ (ie násobí se stupeň příslušnosti k množině A se stupněm příslušnosti k množině B).

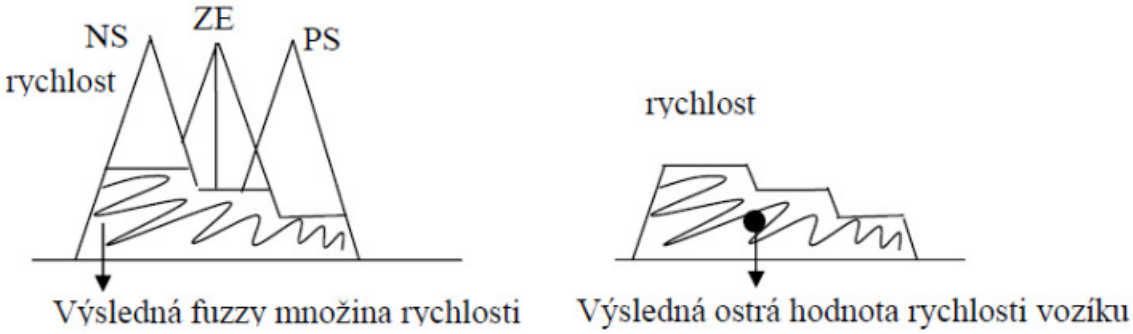
Pravidla mohou mít přiřazeny ještě **váhu** - tou se dodatečně násobí výsledek. *Váhy by imo taky měli dávat v součtu 1, nejsem si 100% jistej, ale dává mi to smysl.*

Fuzzy blok
Viz slajdy:



Obecně, vstupy modelu jsou fuzzifikovány - **fuzzifikace** je převod vstupních hodnota na míry příslušnosti k jednotlivým množinám. Pak následuje proces **inference** - tj. vstupy jsou přivedeny do pravidel a ta se vyhodnotí, výsledky každého se vynásobí jejich vahou. Výsledky všech pravidel jsou agregovány. Agregované výsledky se defuzzifikují - **defuziffikace** je převod z fuzzy hodnot na výstupní hodnotu.

Pozn:
Častým způsobem defuzzifikace je hledání těžiště výsledné agregované fuzzy množiny:



(přejato z opory IMP)
Zde šly na vstup dvě proměnné, prošli si nějakými pravidly a výstupy pravidel se sloučily do toho pravého schodovitového útvaru. K tomu tvaru najdeme těžiště a jeho x-ovou hodnotu vracíme na výstupu.

* Pozn: PP sám upozorňoval - nepleťme si ten interval <0; 1> s pravděpodobností. P-ost říká, jak velká je šance, že se stane X. Fuzzy říká spíš něco jako, jak moc platí X. Hl. rozdíl mi přijde v tom, že v p-osti, "ve 20% platí celé X, v 80% neplatí X vůbec", kdežto u fuzzy "X plati na 20%"...

Credits
Nindaleth (za doplnění příkladů užití... btw odsud)

_____
Součástí každého odborníka v IT, který se zabývá sítěmi, je, že se nebojí číst RFC dokumenty a umí si je najít. -Matoušek Petr, Ing., Ph.D.
They are my space marines and they shall know no fear. -The Emperor of Mankind

Naposledy upravil skeWer dne 01 úno 2011 06:54 pm, celkově upraveno 1

profil sz email upravit citace

Zobrazit příspěvky za předchozí: Všechny příspěvky Seřadit podle Čas odeslání Vzestupně Přejít

nové téma odpovědět **Stránka 1 z 2** [Příspěvků: 18] **Přejít na stránku 1, 2 Další**

Obsah fóra » Předměty (2010/2011) » 3BIT » Zimní semestr » IMS

Kdo je online

Registrovaní uživatelé: bigg.i, Blacker, borek, Conflict, Cospel, DragonLich, Ehm, Ekharion, hacky, HeX, imli, izin, jilji, Kacal, kedysek, literat, ltfatal, marthy, Marty, Paja, skeWer, small, Srnka, st3vou, tomaskolo, Twyer, U.R.A.N., Vlczech, zolex

Můžete zakládat nová témata v tomto fóru
Můžete odpovídat v tomto fóru
Můžete upravovat své příspěvky v tomto fóru
Můžete mazat své příspěvky v tomto fóru
Můžete přikládat soubory v tomto fóru

Hledat: Přejít Přejít na: IMS Přejít