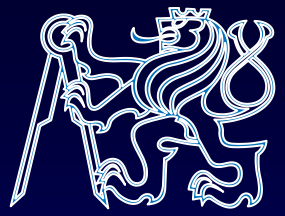
- ©Jan Schmidt 2011, 2013
  Katedra číslicového návrhu
  Fakulta informačních technologií
  České vysoké učení technické v Praze
- Zimní semestr 2013/14



EVROPSKÝ SOCIÁLNÍ FOND
PRAHA & EU: INVESTUJENE
DO VAŠÍ BUDOUCNOSTI

MI-PAA

6. Praktické nasazení algoritmů

- Praktické požadavky a omezení
- Praktické instance
- Práce s heuristikou
- Experimenty: otázky a odpovědi
- Vizualizace

POŽADAVKY A OMEZENÍ

PRINCIPIÁLNÍ POŽADAVEK:

SLUŠNÝ VÝKON V PRAKTICKÉ APLIKACI

- CO JE TO SLUŠNÝ VÝKON
- CO VŠECHNO ZNAMENÁ PRAKTICKÁ APLIKACE

Příklad 1: řešení instalačních závislostí

- **Balíček:** atomická instalační jednotka
  - má **jméno** a **verzi**
  - poskytuje pojmenované **schopnosti** (capabilities), které také mohou mít verzi
- **Závislost:** vztah mezi balíčky
  Balíček vyžaduje ke správné funkci množinu schopností
- **Konflikt:** vztah mezi balíčky
  Balíček vyžaduje ke správné funkci **nepřítomnost** množiny schopností
- **Repozitář:** množina balíčků a závislostí a konfliktů mezi nimi

Instalace jako problém splnitelnosti

Máme k dispozici obsah **repozitáře**, všechny závislosti a konflikty

Instalace	**SAT**
Instalace baličku A	proměnná a
Dotaz na instalaci baličku A	hodnota a v řešení
Požadavek na instalaci baličku A	(a)
Baliček A **závisí** na schopnosti poskytovane baličkem B.1 nebo B.2 (zapomeň na A nebo instaluj B.1nebo instaluj B.2)	$(\neg a + b1 + b2)$
Baliček B **koliduje** s baličkem C (zapomeň na B nebo zapomeň na C)	$(\neg b + \neg c)$

Nalézt ohodnocení proměnných (=rozhodnutí o instalaci) takové, že hodnota formule je 1 (=systém je konzistentní a požadavky jsou splněny)

Požadavky

- **Rychlost:** čas řešení je dobou odezvy
- **Kvalita:** exaktní algoritmus
- **Optimalizace:** na velikost stažení, aktuálnost systému
- **Skrytost:** uživatel nemá vědět, co je to SAT
- **Relaxace:** najít minimum změn, které učiní formuli splnitelnou
- **Vysvětlení:** poskytnout důvody pro nesplnitelnost formule a potřebné změny v **pojmech balíčků, závislostí** atd.

```
# zypper install fortran
mi-paa-zypper: invoking Las Vegas solver
mi-paa-zypper: set learning instensity: 0.78
mi-paa-zypper: set watchlist length: 62
mi-paa-zypper: set var heuristic: minsat
mi-paa-zypper: estimated time 72hrs
mi-paa-zypper: remaining 71h 59m 59s
mi-paa-zypper: conflict in clause 3352
mi-paa-zypper: still trying …
mi-paa-zypper: conflict in clause 88671
mi-paa-zypper: still trying …
mi-paa-zypper: giving up
```

Příklad 2: verifikace kontrolou modelu

- Model: vlastnost (výrok), vztažený k hodnotám proměnných a kroku výpočtu (času)
- Jedna z možných technik:
  - popis kroku výpočtu CNF
  - zkopírování pro *k* následujících kroků
  - popis vlastnosti CNF
  - SAT solver

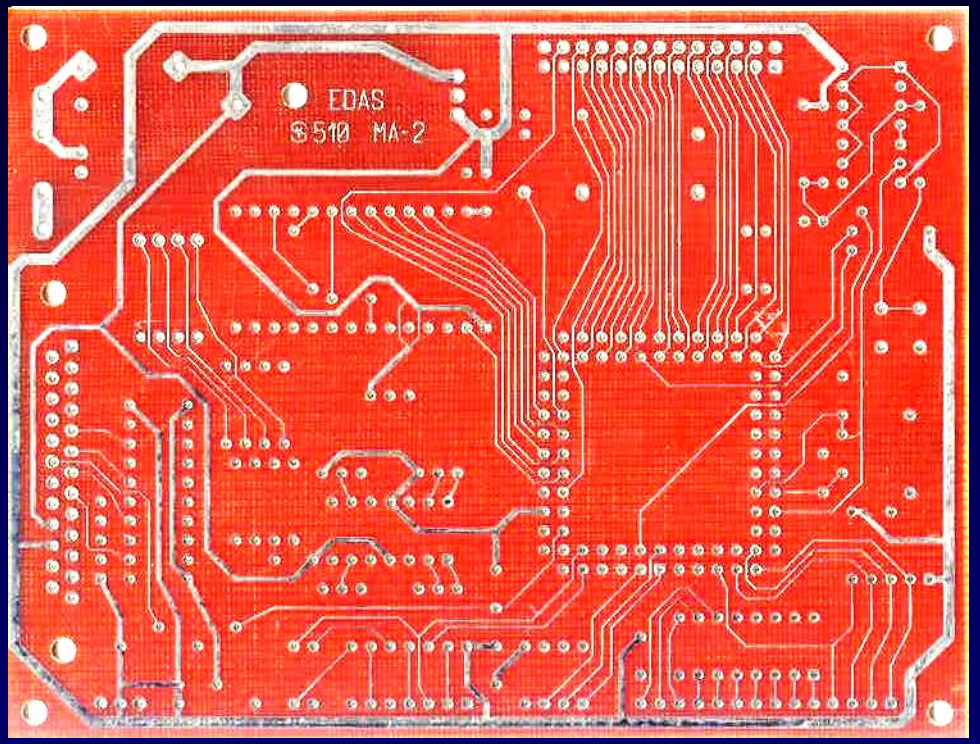
Požadavky

- **Rychlost:** podle důležitosti, i několik dní
- **Kvalita:** exaktní algoritmus
- **Skrytost:** uživatel nesmí vědět, co je to SAT ani jak pracují převody z výpočtu na CNF a z popisu vlastnosti na CNF – podmínka průmyslového využití
- **Vysvětlení:** poskytnout **protipříklad** – posloupnost vstupů, která vede k porušení zadané vlastnosti

PRAKTICKÉ INSTANCE

Drahý šéfe,

jak jste psal, že to Václav testoval na souborech náhodně vygenerovaných bodů a že to mám sofort hned přepsat, tak jsem to otestoval na deskách od nás a i z vedlejšího oddělení a funguje to zcela úplně dobře. Pravděpodobný důvod najdete na přiloženém obrázku. Algoritmus si z řad blízkých bodů vytvoří „stavební bloky“ a pak už je to snadné.



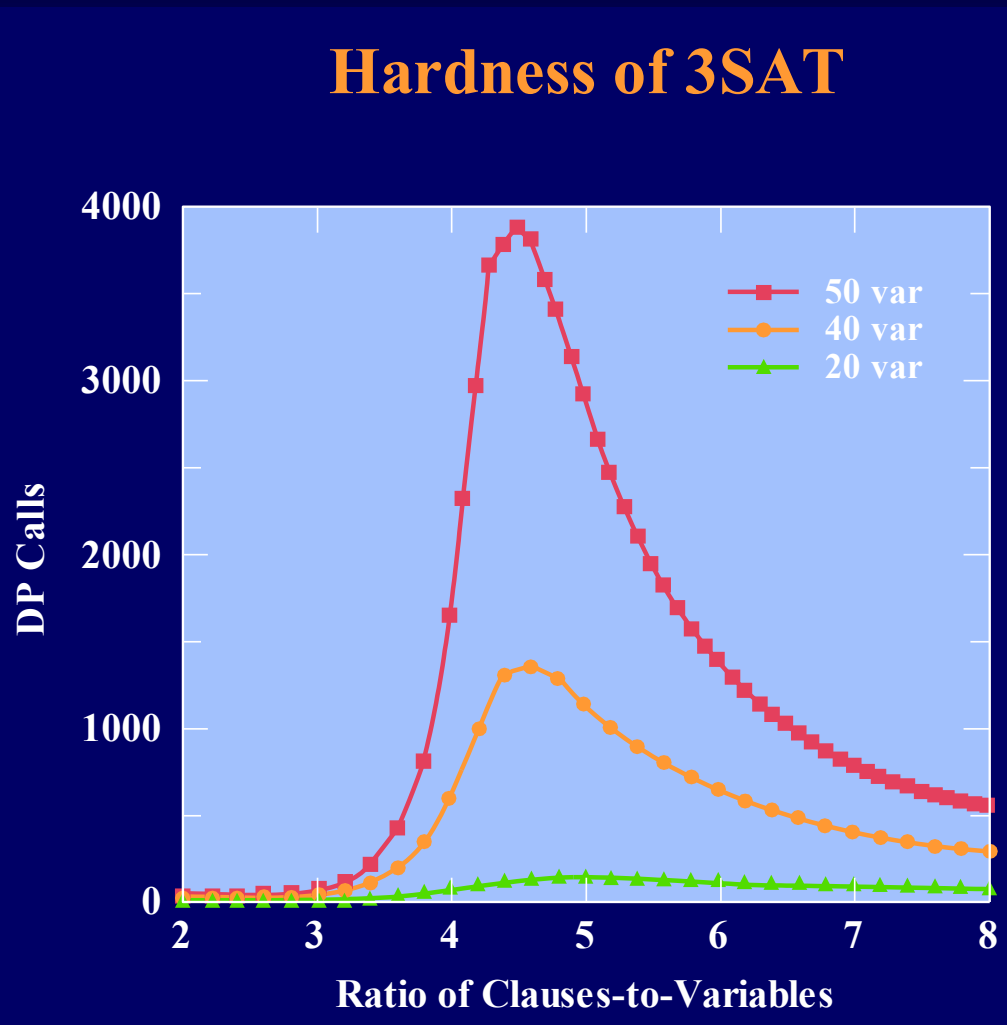
Tak jsem nic nepředělával. Srdečné pozdravy z Bali. Pozdravujte Václava.

Není SAT instance jako SAT instance

- Problém instalačních závislostí: pokud je každá schopnost poskytována nejvýše jedním balíčkem,
  - vzniknou tzv. **Hornovy klauzule**
  - takové instance jsou řešitelné v polynomiálním čase
- 2-SAT je řešitelný v polynomiálním čase
- Problém automatického generování testu číslicového obvodu:
  - průměrný počet literálů je okolo 2.3, tzv. 2+p-SAT (2 a kousek)
  - ve **slušných** řešičích se chová jako 2-SAT
- Problém fázového přechodu

Fázový přechod

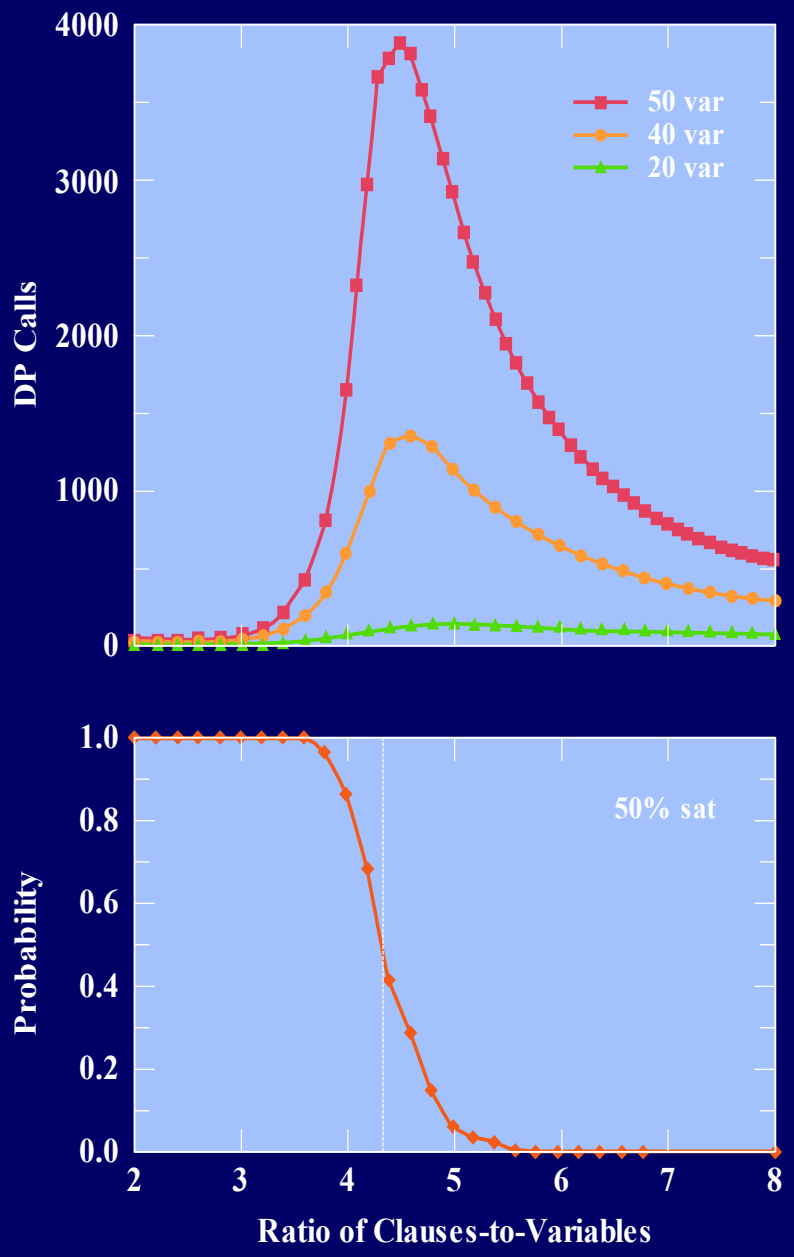
Stochastic Search
And Phase Transitions:
AI Meets Physics
Bart Selman
AT&T Bell Laboratories
Murray Hill, N.J. USA
www.cs.cornell.edu/home/
selman/papers-ftp/
ai-phys1.ppt



Závislost počtu kroků
procedury Davis-Putnam
na poměru
počtu klauzulí
k počtu proměnných

Fázový přechod

- Pravděpodobnost splnitelnosti v závislosti na poměru počtu klauzulí k počtu proměnných
- Analogie se skupenským teplem fázového přechodu látky
- Snadno řešitelné instance řešeny rychle
- Neřešitelné instance rozpoznány rychle



Booleovy funkce

- Shannon 1949:
  Jestliže $n \to \infty$, podíl n-árních funkcí s obvodovou složitostí méně než $2^n/3n$ klesá k 0.
- Booleovy funkce, popisující reálné obvody, takové nejsou (platilo za Shannona, platí dodnes)
- JAK TO?
- A na čem mám zkoušet svoje heuristiky?

Standardní sady zkušebních instancí (benchmarks)

- **Veřejně dostupné sady instancí**
  - vztažených k určité úloze
  - dovolující porovnávání algoritmů (publikované výsledky)
- **Podle účelu**
  - praktické algoritmy: instance z praxe
  - teoretická měření : speciálně konstruované
  - ISCAS, MCNC, ITC: obvody
  - DIMACS: SAT
  - TSP sady

Příklad: ISCAS – kombinační obvody

charakteristiky

Circuit Name	Circuit Function	Total Gates	Input Lines	Output Lines	Faults[1]
C432	Priority Decoder	160 (18 EXOR)	36	7	524
C499[2]	ECAT	202 (104 EXOR)	41	32	758
C880	ALU and Control	383	60	26	942
C1355[2]	ECAT	546	41	32	1574
C1908	ECAT	880	33	25	1879
C2670	ALU and Control	1193	233	140	2747
C3540	ALU and Control	1669	50	22	3428
C5315	ALU and Selector	2307	178	123	5350
C6288	16-bit Multiplier	2406	32	32	7744
C7552	ALU and Control	3512	207	108	7550

[1]Reduced equivalent fault set based on equivalence fault collapsing.
[2]Circuits C499 and C1355 are functionally equivalent. All EXOR gates of C499 have been expanded into their 4-NAND gate equivalents in C1355.

orientace sady (testování)

EXPERIMENT

Charakterizace algoritmu

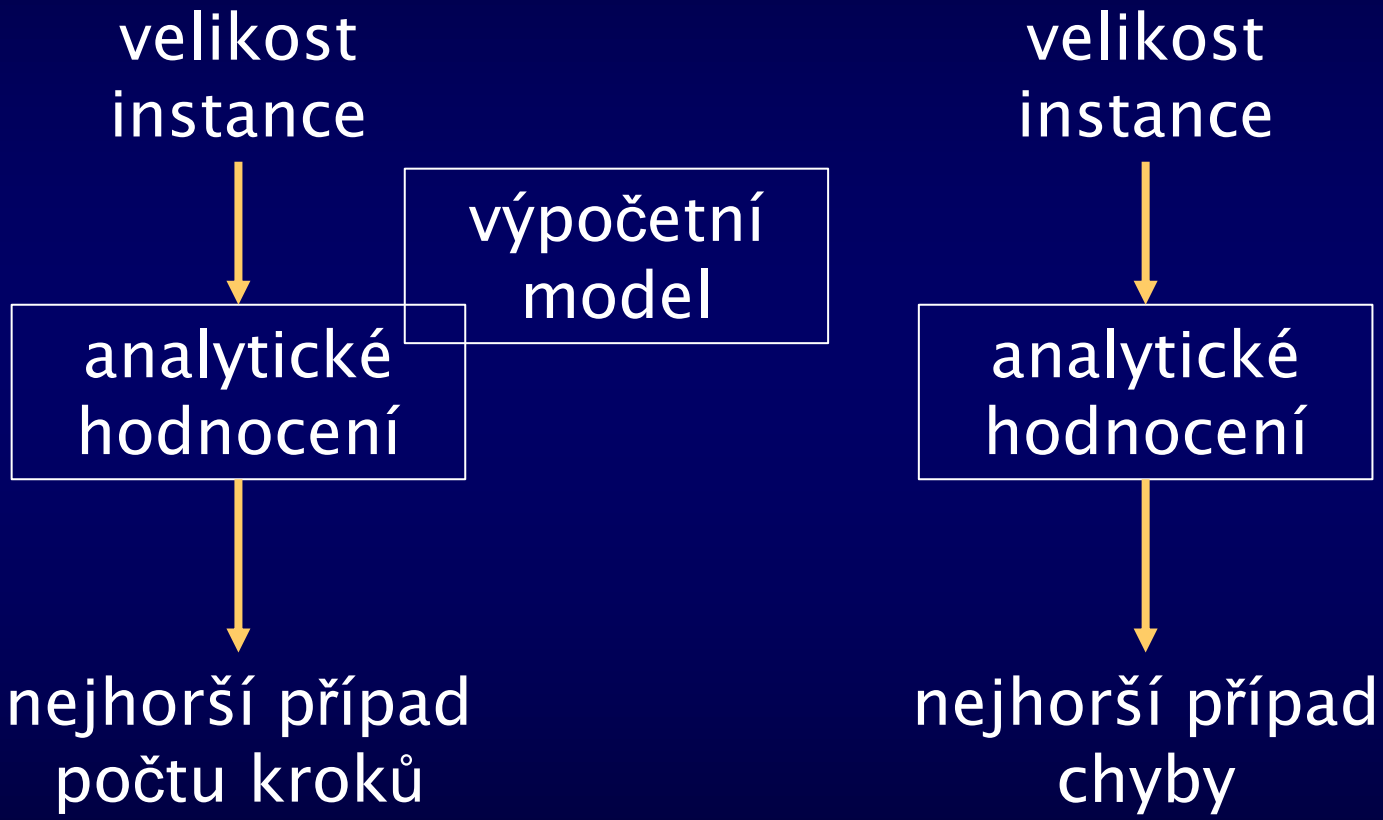
- **Účel:** předpovědět, jak bude algoritmus řešit libovolnou instanci s přijatelnou nejistotou
- **Problém:** čím charakterizovat práci algoritmu
  - počet kroků
  - spotřeba paměti
  - heuristiky: kvalita řešení

Charakterizace instance

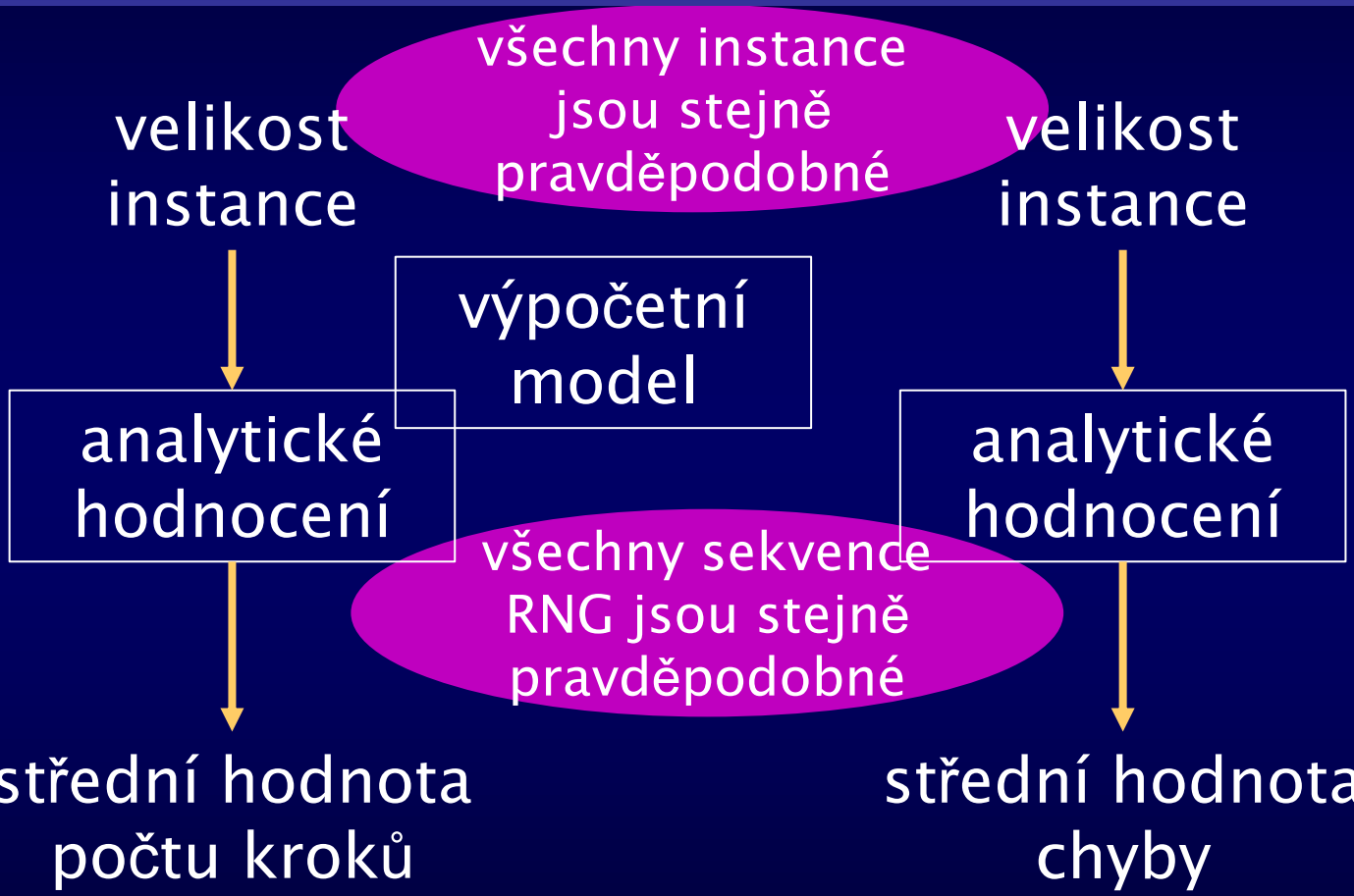
- Pracovat s úplnou charakterizací instance (=hodnoty vstupních proměnných) znamená instanci vyřešit
- Problém: čím charakterizovat instanci
  - hrubá velikost
  - jemná velikost
  - …
  - …
  - komunikační složitost???
  - …

víme, že na 3SAT to moc nefunguje (= nedává spolehlivou předpověď)

Postupy charakterizace
deterministické algoritmy, nejhorší případ

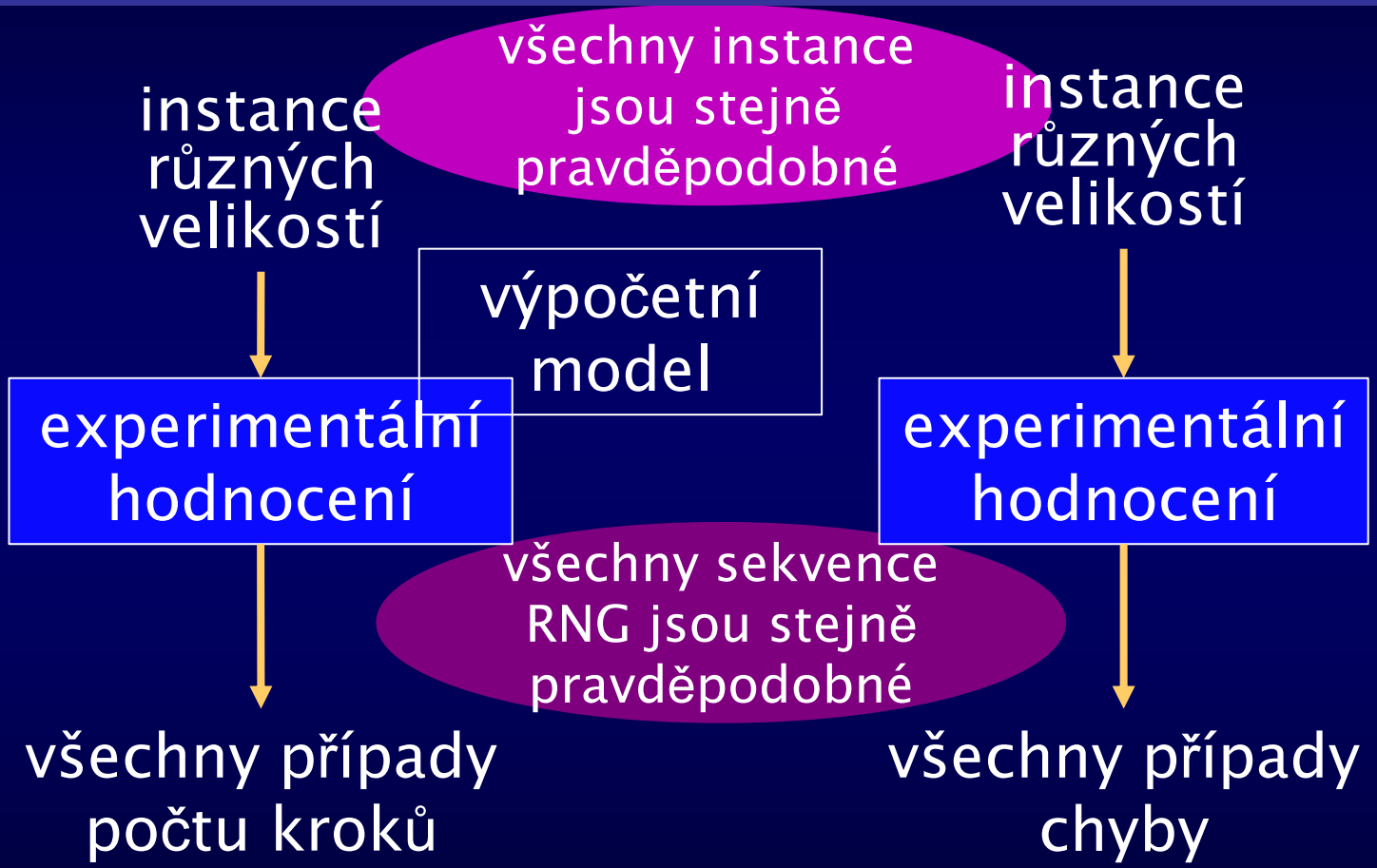


Postupy charakterizace
randomizované algoritmy, očekávaný případ



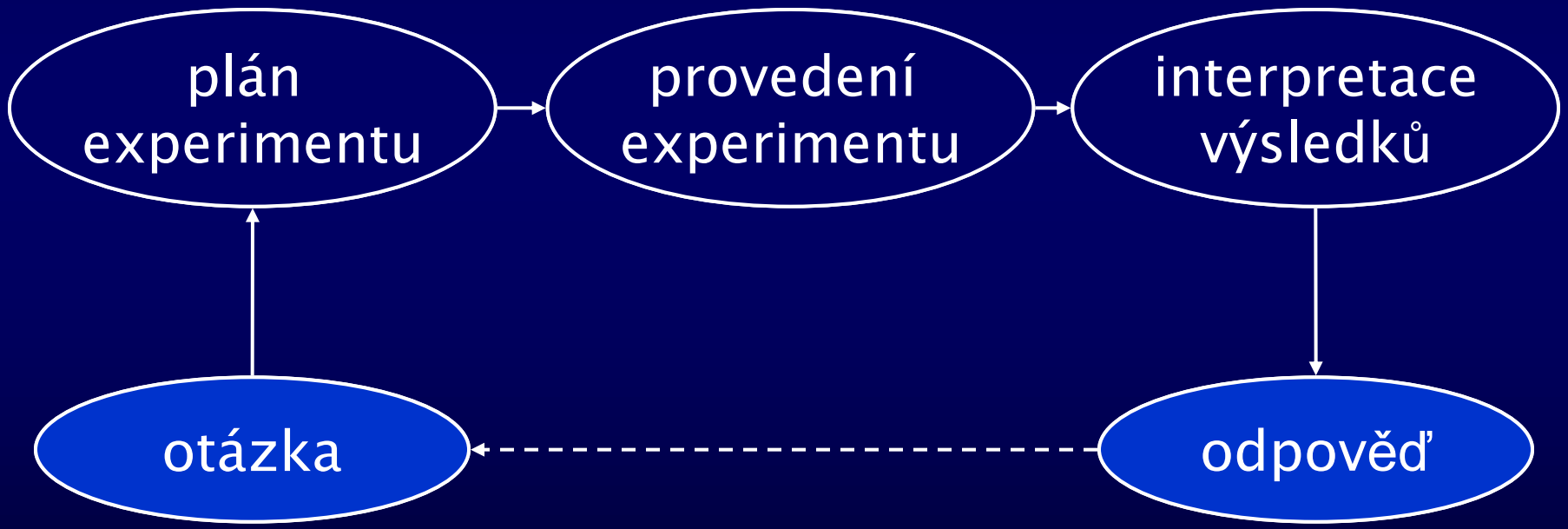
ale už jednoduché algoritmy je těžké analyzovat

Experimentální algoritmika
očekávaný případ

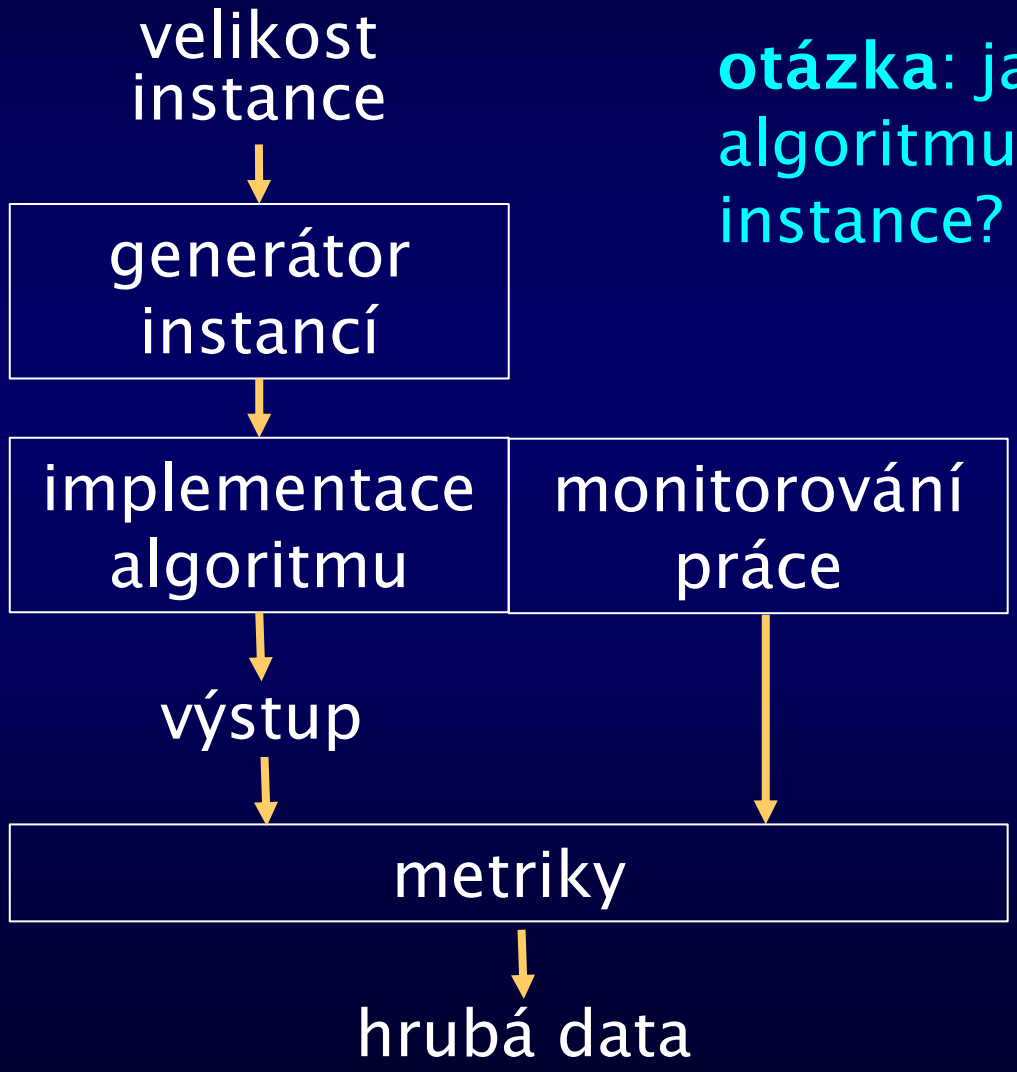


Experiment

- statistická
- vizuální
- …

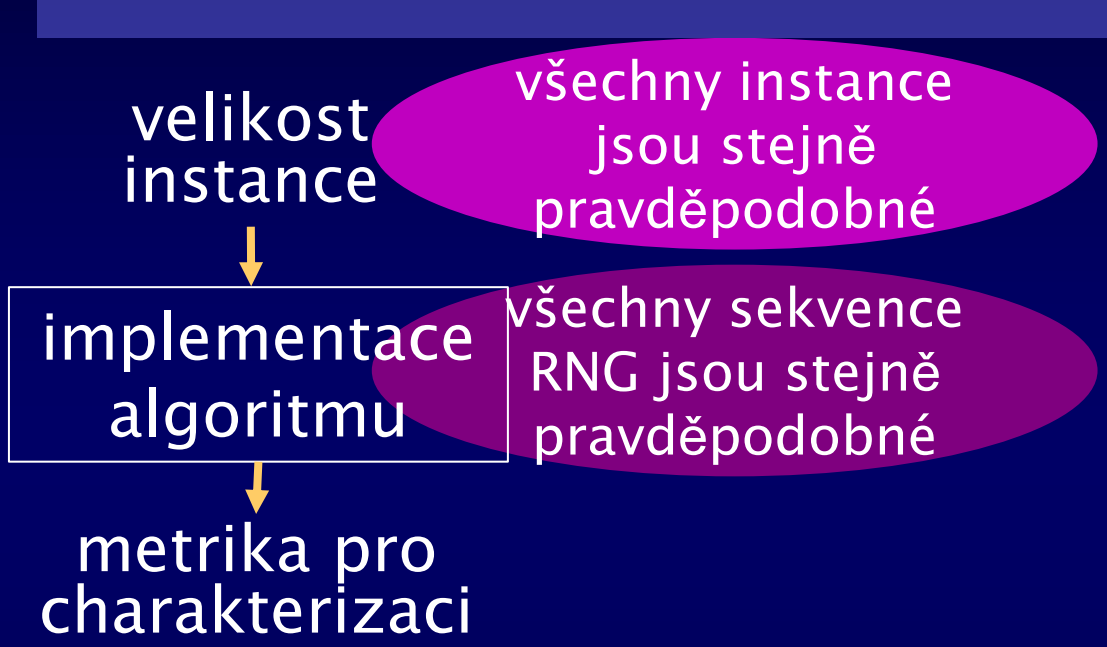


Experimentální algoritmika



otázka: jak závisí práce algoritmu na charakterizaci instance?

Experimentální algoritmika
vyhodnocení – závislost na velikosti instance



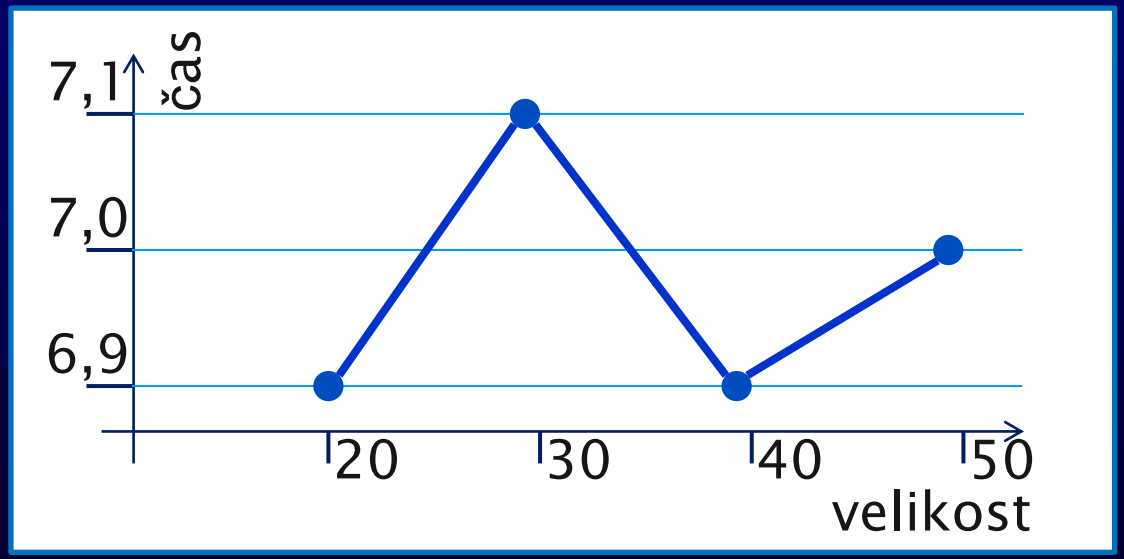
zajímá nás jen velikost instance; všechno ostatní je „šum“

randomizovaná povaha algoritmu vytváří „šum“

- O šumu platí statistické předpoklady
- Pro odstranění šumu užijeme statistických metod
- Postupně neutralizujeme jednotlivé zdroje šumu
- Např. pro každou instanci měříme vícekrát

Experimentální algoritmika
kritické zhodnocení

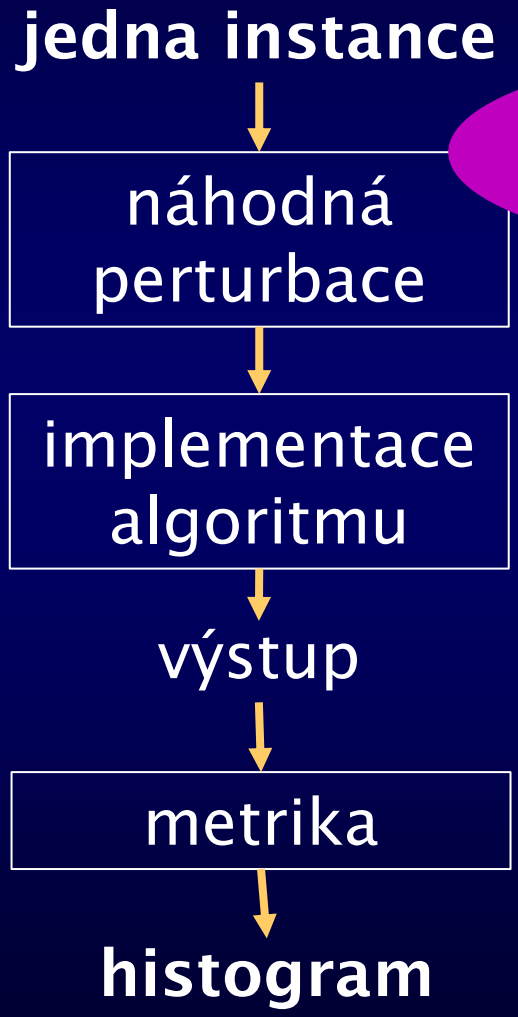
- Jsou naměřené hodnoty spolehlivé? (statistická kriteria) → dostatečný počet měření na jedné instanci?
- Jsou naměřené hodnoty vysvětlitelné? (Nemonotonní závislosti, skoky …)
- Ne-li, jaké experimenty je třeba dále provést?



Porozumění

- Výstup z experimentu: kvantitativní data
- Vysvětlit znamená porozumět
- Vysvětlit: nalézt zákonitost
- Zákonitost: podívám se a vidím…
  - histogramy
  - bodové grafy
  - logaritmické měřítko
- Když uvidím: matematické modelování, statistika…
- Kvalitativní informace
- Porozumění

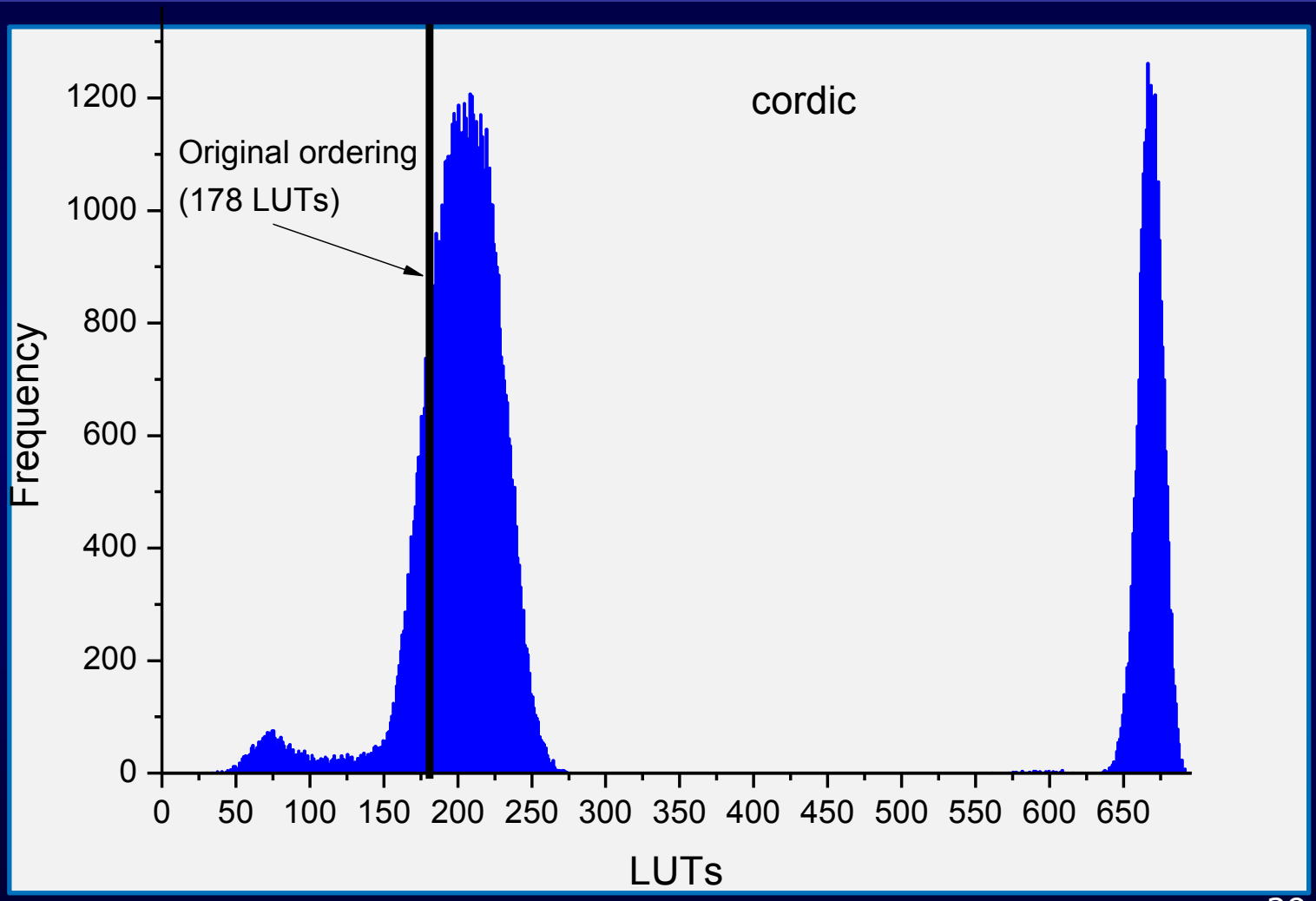
Měření robustnosti heuristiky
(odolnosti proti perturbacím vstupního popisu)



všechny perturbace jsou stejně pravděpodobné

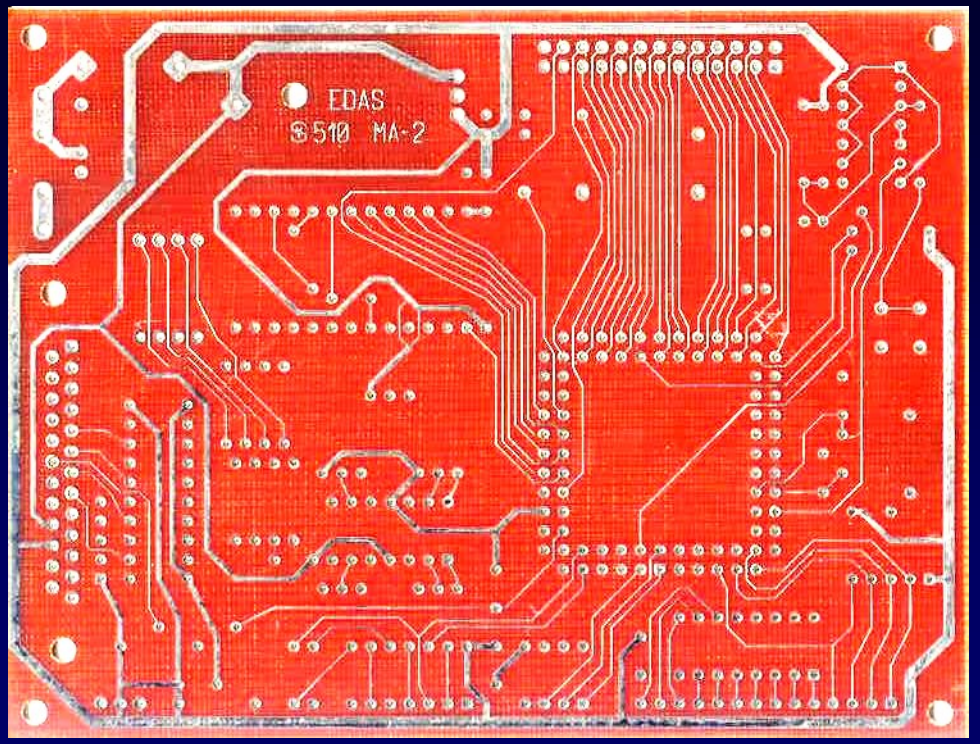
- Robustnost:
  - zpřeházím klauzule SAT solveru a dostanu jiné řešení v jiném čase
  - zpřeházím pořadí deklarací proměnných v programu a ten funguje rychleji

Histogram kvality řešení
(Petr Fišer, minimalizční problém)



Zhodnocení experimentu

- Podívám se a vidím: ten algoritmus si dělá co chce
- Z tohohle počítat průměr??
- Porozumění: co znamenají vrcholy histogramu?
- Teorie: shluky suboptimálních řešení
- Statistický model: suma normálních rozložení
- …
- …
- *dva roky práce*
- …
- **daná instance je pro praxi irelevantní**

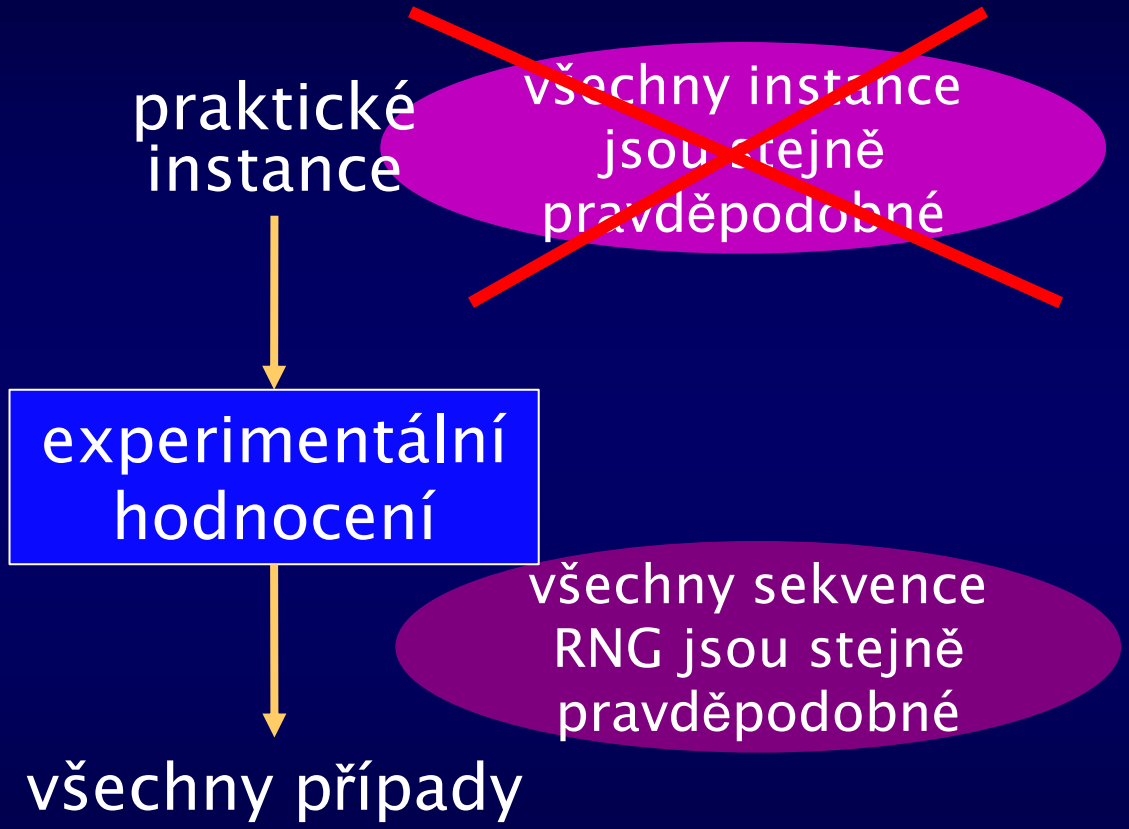


Drahý šéfe,

jak jste psal, že to co jsem Vám psal o experimentálním vyhodnocení, nemá vůbec žádný vědecký základ, tak tomu úplně tak není.

Je jenom třeba nalézt způsob, jak se vědecky charakterizuje, co je to „deska od nás“ a „deska z vedlejšího oddělení“. Pak to půjde docela snadno. Co je to praktický obvod, zkoumal už roku 1949 nějaký Shannon. Dodnes se to pořádně neví.

Inženýrská experimentální algoritmika



- Často neumíme charakterizovat ani co je to „praktická instance“, tím méně je generovat

Standardní sady zkušebních instancí (benchmarks)

- **Veřejně dostupné sady instancí**
  - vztažených k určité úloze
  - dovolující porovnávání algoritmů (publikované výsledky)
- **Podle účelu**
  - praktické algoritmy: instance z praxe
  - teoretická měření : speciálně konstruované
  - ISCAS, MCNC, ITC: obvody
  - DIMACS: SAT
  - TSP sady

Příklad: ISCAS – kombinační obvody

charakteristiky

Circuit Name	Circuit Function	Total Gates	Input Lines	Output Lines	Faults[1]
C432	Priority Decoder	160 (18 EXOR)	36	7	524
C499[2]	ECAT	202 (104 EXOR)	41	32	758
C880	ALU and Control	383	60	26	942
C1355[2]	ECAT	546	41	32	1574
C1908	ECAT	880	33	25	1879
C2670	ALU and Control	1193	233	140	2747
C3540	ALU and Control	1669	50	22	3428
C5315	ALU and Selector	2307	178	123	5350
C6288	16-bit Multiplier	2406	32	32	7744
C7552	ALU and Control	3512	207	108	7550

[1]Reduced equivalent fault set based on equivalence fault collapsing.
[2]Circuits C499 and C1355 are functionally equivalent. All EXOR gates of C499 have been expanded into their 4-NAND gate equivalents in C1355.

orientace sady (testování)

Inženýrská experimentální algoritmika

- Žádná známá standardní sada instancí nemá zaručenu statistickou reprezentativnost
- Neplatí tedy statistické předpoklady
- Nefunguje ani průměrování
- Nemůžeme eliminovat neznámé zdroje šumu
- Ze statistického aparátu nám zbyl jen existenční kvantifikátor (existuje instance, na které …)

- A jak je takový výrok relevantní pro praxi?
- Ptejte se autora sady

PRÁCE S HEURISTIKOU

Parametry, metaheuristiky

zásuvné moduly

omezená indikace

množství parametrů

- různé možné konstrukce určitých částí heuristiky
- číselné parametry
  - nesrozumitelné koncovému uživateli
  - navzájem se ovlivňují
- indikace: hodnota optimalizačního kritéria, pokud si neopatříme něco lepšího

„Odborná obsluha“ heuristiky

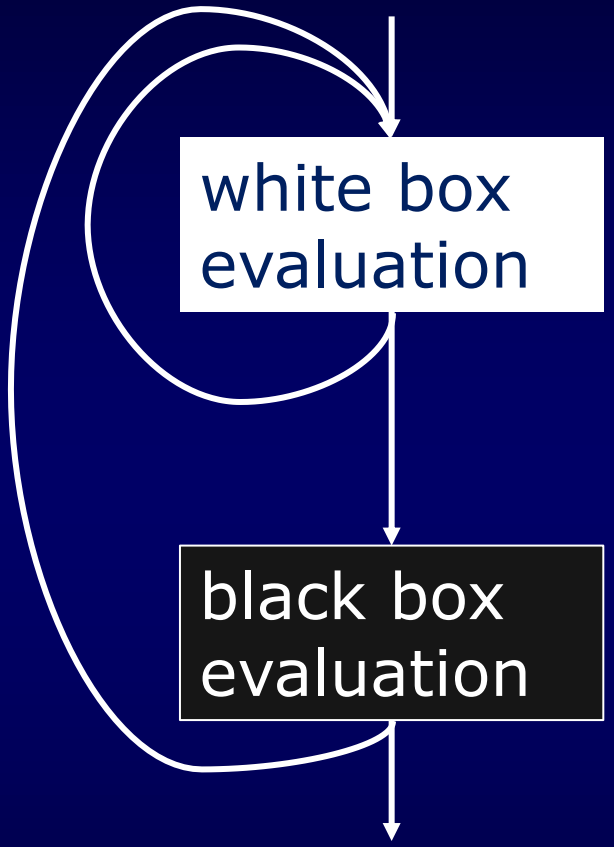
- **Obsluha nerozumí** funkci přístroje, **neví** účel ovládacích prvků:
  - zkusí měnit jednotlivé parametry, pro každý zaznamená hodnotu opt. kritéria („sílu signálu“)
  - nastaví všechny prvky do zjištěných optim
  - nebo vyhledává optima postupně
  - dokáže „chytit místní stanici“
- **Obsluha rozumí** funkci:
  - zajistí, aby „přístroj“ pracoval správně
  - sleduje charakter práce
  - dokáže měřit další veličiny, nutné k porozumění stavu „přístroje“
  - využije všech schopností „přístroje“

Praktická aplikace: jen dvě možnosti

- **Nastavit knoflíky jednou provždycky**
  a dokázat, že to bude „hrát“ pro **všechny** praktické instance
- **Udělat robota,** který
  - se podívá na instanci a nastaví knoflíky
  - poslouchá, jak to „hraje“ a upravuje nastavení podle vlastního algoritmu

  a dokázat, že to bude „hrát“ pro **všechny** praktické instance

Práce s heuristikou



- omezená sada instancí
- detailní měření
- vhled, porozumění
- modifikace heuristiky

- plná sada instancí
- měření výsledků
- ověření kvality a výkonu
- žádné modifikace heuristiky

Problém: odlišit ne dost dobře nasazenou heuristiku od **programátorské chyby**

Otázky

- Je algoritmus A lepší než algoritmus B?
- Pro praktické instance, je algoritmus A lepší než algoritmus B?
- Pro praktické instance, reprezentované těmito zkušebními úlohami, je algoritmus A lepší než algoritmus B?

Odpovědi a jejich kvalifikace

- Je algoritmus A lepší než algoritmus B?
- Některé charakteristiky algoritmu A na některých instancích jsou lepší než B, jindy opačně
- Pro praktické instance, je algoritmus A rychlejší než algoritmus B?
- Lze-li považovat úlohy za statisticky významné, pak ano pro malé instance (do 10 000 prvků)
- Pro praktické instance, reprezentované těmito zkušebními úlohami, je algoritmus A lepší než algoritmus B?
- A dává přesnější výsledky, ale pro velké instance (nad 10 000 prvků) je pomalejší

Vývojářovy otázky

- Algoritmus:
  1. Počáteční ohodnocení Y: každou proměnnou ohodnoť 0 nebo 1 se stejnou pravděpodobností.
  2. S pravděpodobností $0<q<1$ proveď 3 jinak proveď 4.
  3. Najdi ohodnocení Y', které se liší od Y v právě jedné proměnné některé nesplněné klauzule
  4. Najdi ohodnocení Y', které se liší od Y v právě jedné proměnné a poskytne nejvíce splněných klauzulí
  5. $Y \leftarrow Y'$. Pokud nejsou všechny klauzule splněny nebo vyčerpán stanovaný počet kroků, opakuj 2.

co tam mám napsat, aby to počítalo?

… a odpovědi

co tam mám napsat, aby to počítalo?

Experiment:
měním q a ověřuji pro reprezentativní sadu instancí

mně to nechodí! vždyť je to pokaždé jinak!

Experiment:
měním charakteristiky instancí a ověřuji činnost adaptačních algoritmů nastavujících q

jaké charakteristiky instance ovlivňují q?

→ další otázky, další experimenty

Více parametrů

- Parametry obecně nejsou nezávislé!
- U některých parametrů je vzájemná závislost známa, nutno respektovat.
- Nezávislost parametrů, není-li známa, nutno ověřit $\rightarrow$ otázky, odpovědi.
- Nastavování více parametrů je cesta prostorem konfigurací heuristiky!
- Znalosti, vhled

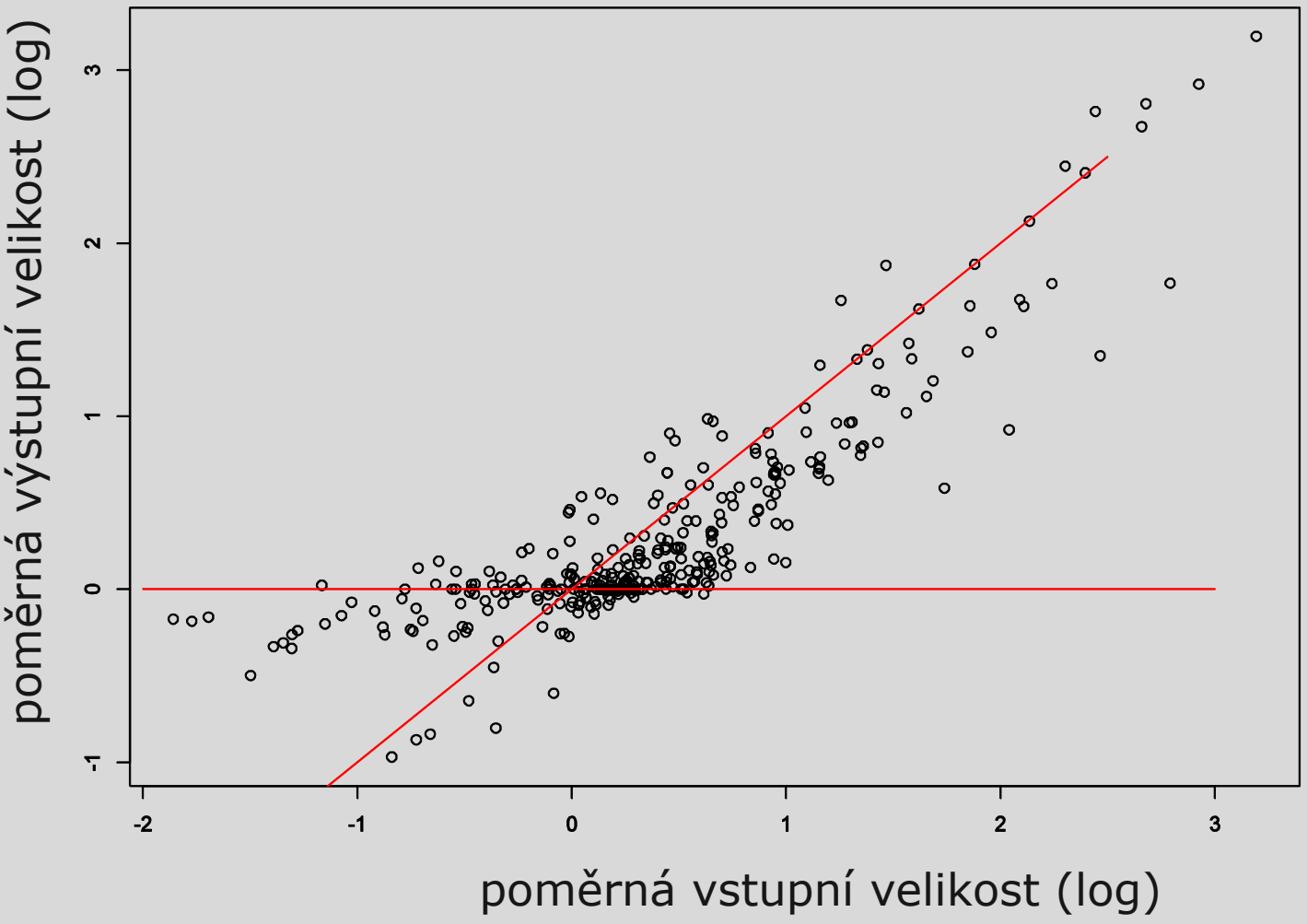
Vizualizace

- Pro heuristiky, které pracují **iterativně s jednou aktuální konfigurací**, nejčastěji:
  vývoj optimalizačního kritéria s pořadím iterace
- **Podívám se a vidím**
  - heuristika se vrhne na nejbližší trochu dobré řešení a jiné nezkoumá
  - heuristika bezcílně bloudí prostorem konfigurací
  - heuristika nachází čím dál tím horší řešení
  - heuristika hledá správně, ale výpočet byl ukončen brzy

Vizualizace výkonu heuristiky

- Vstup: popis Booleovy funkce
- Výstup: minimální popis Booleovy funkce
- Pro stále stejnou funkci:
  - zvětším nebo zmenším originální „průmyslový“ vstupní popis (určitým netriviálním způsobem)
  - zvětší nebo i zmenší se výstupní popis? Neměl by – funkce se nezměnila
  - udělám pro několik set instancí
- *…*
- *… a jak že je to s tou relevancí?*

Vizualizace výkonu heuristiky



bodový graf (korelační diagram, scatter plot)

Praktické nasazení heuristik

- Praktické požadavky
  - rychlost , kvalita
  - na souboru instancí, jehož významné vlastnosti neumíme charakterizovat
  - žádná intervence koncového uživatele
- Heuristiky
  - výměnné části, parametry
  - vzájemná závislost
  - často iterativní charakter
- Práce s heuristikou a experimentální vyhodnocení
  - vhled do činnosti heuristiky
  - white box, black box
  - standardní zkušební úlohy
  - otázka, plán, experiment
- Vizualizace: podívám se a vidím