Experimentální vyhodnocení algoritmu

NI-KOP // Ondřej Wrzecionko

04.11.2022

Zadání

Experimentálně srovnejte algoritmy GSAT a probSAT. Určete, který algoritmus dospěje rychleji (v menším počtu iterací) k řešení obtížných instancí 3-SAT v rozsahu 20-75 proměnných. Zdůvodněte použité metody a metriky, popište interpretaci dat.

Uvažujte pevné parametry:

- GSAT: $p = 0.4$
- probSAT: $c_m = 0, c_b = 2.3$

Abstrakt

V 1. domácím úkolu jsem se zabýval experimentálním srovnáním algoritmů gSAT a probSAT, které řeší problém splnitelnosti Booleovské formule. Vygeneroval jsem obtížné instance 3-SAT s 20 až 75 proměnnými, vyřešil jsem je s pomocí těchto algoritmů, následně jsem zanalyzoval různé metriky a vybral jsem nejvhodnější, s pomocí kterých jsem srovnal tyto algoritmy. Na základě všech metrik jsem dospěl k závěru, že probSAT na daných instancích jasně dominuje nad gSATem.

1 Úvod

Máme **dva algoritmy** – gSAT a probSAT a zabýváme se tím, který z nich je efektivnější při řešení obtížných instancí 3-SAT v rozsahu 20-75 proměnných.

Obtížná instance problému 3-SAT je v našem rozsahu proměnných taková, kde je poměr počtu klauzulí ku počtu proměnným přibližně 4.3.

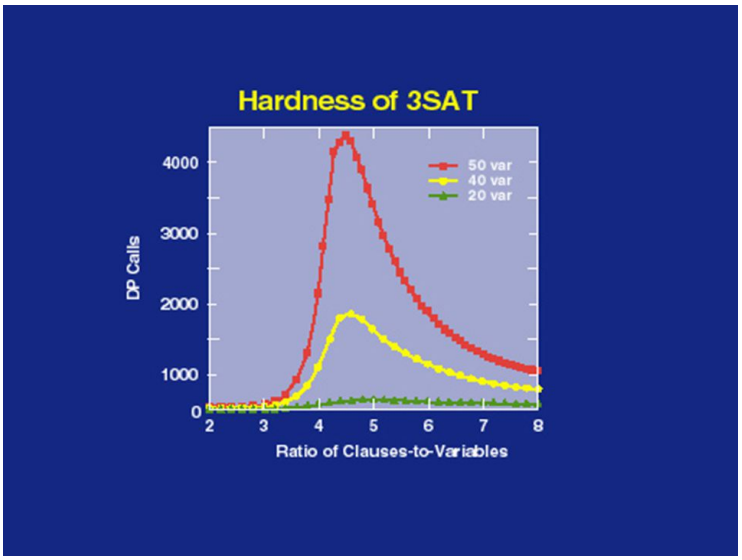

Figure 1: Graf obtížnosti 3-SAT na základě poměru

Mým cílem je prozkoumat chování obou dvou algoritmů při řešení těchto obtížných instancí a na základě jednotlivých metrik pak určit, zda některý z nich **dominuje** nad druhým.

2 Materiál

2.1 Generace instancí

Nejprve jsem **vygeneroval** obtížné instance 3-SATu. K tomu jsem použil generátor Power Law Random SAT Generator.

```
NUMOFVARS=$(( 20 + ( RANDOM % 56 ) ))
NUMOFCLAUSES=$(( 42 * NUMOFVARS / 10 ))

./CreateSAT -q -g u -c $NUMOFCLAUSES -v $NUMOFVARS -k 3 -f
"inst/gen-$i"
```

Listing 1: Shell příkaz pro generaci instancí

První řádek vygeneruje náhodně počet proměnných v rozsahu 20 až 75 proměnných, druhý následně vygeneruje počet klauzulí jako počet proměnných krát 4.2. Je tedy zaručeno, že se bude jednat o **obtížné** instance 3-SAT *(viz. Úvod)*.

Při generaci používám přepínač **-q**, který způsobí, že se nevypisují pomocné informace, **-g u**, který způsobí generaci proměnných pomocí rovnoměrného rozložení, **-c** a **-v**, které nastaví počet klauzulí a proměnných, **-k 3**, které definuje 3-SAT (3 proměnné v jedné klauzuli) a **-f**, které i-tou instanci uloží vždy do složky inst jako soubor **gen-i.cnf**.

2.2 Algoritmy

Pracuji s algoritmem **gSAT**, který byl popsán na 3. cvičení a jehož implementace je dostupná ke stažení na stránkách předmětu.

Dále pracuji s algoritmem **probSAT**, který byl popsán na stránce zadání úkolu. Konkrétní implementace nám v rámci předmětu nebyla dodána, proto jsem použil mírně upravenou implementaci z Githubu, která odpovídá té popsané v zadání úkolu.

```
for j in `seq 1 1000`
do
  ./gsat2 -r time -p 0.4 -i 1500 "inst/gen-$i.cnf" > /dev/null
  2>> "inst/gsat-$i.dat" &
  ./probsat -r time --cb 2.3 -m 1500 "inst/gen-$i.cnf" >
  /dev/null 2>> "inst/psat-$i.dat" &
done
```

Listing 2: Spuštění gSATu a probSATu na dané instanci

Každý algoritmus spustím pro každou instanci **1000x** se zadanými parametry podle zadání ($p = 0.4$ u gSATu, $c_b = 2.3$ u probSATu), přepínačem **-r** pro pseudo-náhodné řešení, maximálním počtem 1500 iterací a výsledek uložím opět do složky inst/ do souboru s názvem **gsat-i.dat**, resp. **psat-i.dat**.

2.3 Metriky

Pro srovnání algoritmů jsem měl na výběr z následujících metrik:

- počet **úspěšných** běhů, kdy bylo nalezeno řešení daným algoritmem
- **průměrný** počet **iterací** algoritmu přes všechny běhy
- **vážený** počet **iterací** algoritmu, kde se neúspěšný běh počítá jako 10x maximální počet iterací *(Penalized Average Runtime)*
- odhady parametrů **lognormálního** rozdělení pro daný algoritmus
- **xing** – krok, kdy se naposled protly pravděpodobnosti úspěšného řešení *(korigované CDF)* algoritmů
- **winner** – algoritmus, jehož pravděpodobnost úspěšného řešení leží po posledním průsečíku výše

Všechny tyto metriky jsem pomocí shell skriptu (viz. soubor **generate.sh**) spočítal a uložil do souboru **result.dat** ve stejném formátu, jaký byl použit v datech z 4. cvičení.

Skript jsem tedy spustil, vygeneroval 300 instancí problému 3-SATu, vyřešil je, a **výsledný soubor** importoval do Excelu, kde jsem prozkoumal, které metriky budou vhodné k porovnání.

První věc, které jsem si všimnul je, že některé vygenerované instance jsou tak těžké, že je nedokázal vyřešit **ani jeden** algoritmus v ani jednom běhu *(počet úspěšných běhů byl 0, průměrný počet iterací 1500, vážený počet iterací 1500, xing 0)*. Tyto instance jsem tedy **odfiltroval**.

Následně jsem se podíval na možné sledované metriky a všechny z nich, až na odhady parametrů lognormálního rozložení byly použitelné.

Jako **sledované metriky** jsem tedy vybral počet úspěšných běhů, průměrný počet iterací, vážený počet iterací a xing / winner.

3 Výsledky

Skript jsem tedy spustil pro 10 000 instancí. Výsledek skriptu je dostupný v příloze jako soubor (**result.dat**). Tato data jsem následně importoval do Excelu a odfiltroval neřešitelné instance, čímž vznikl soubor **result-filtered.dat**. Následně jsem spočítal průměr počtu běhů / iterací, nebo četnost výhry jednotlivých algoritmů. Výsledná tabulka je dostupná v příloze (**result.xslx)**.

4 Diskuse

4.1 Potlačení statistických odchylek

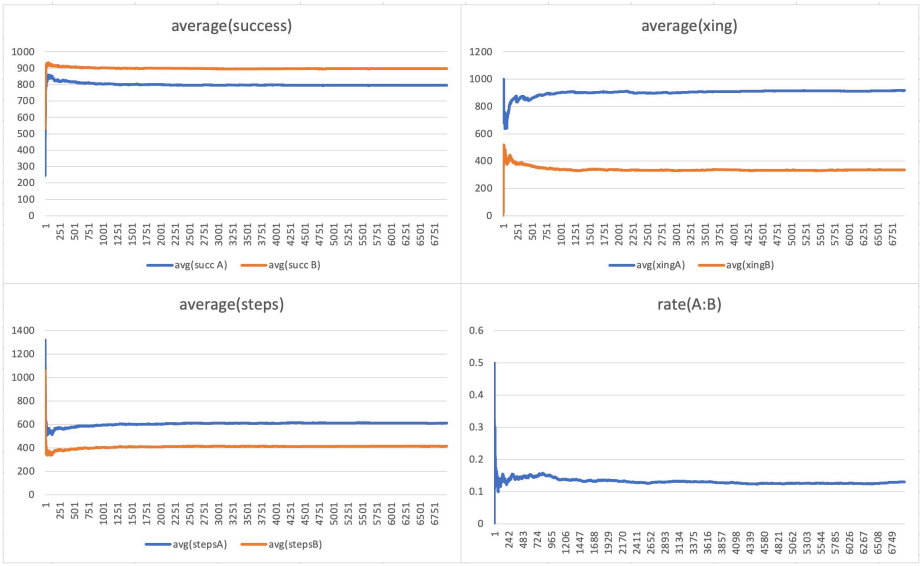

Figure 2: Vývoj průměrného počtu úspěšných běhů, iterací, váženého počtu iterací, průsečíku a poměru výher gSAT ku probSATu. A = gSAT, B = probSAT

Ještě předtím, než jsem z výsledků něco vyvozoval, jsem si s pomocí grafů výše ověřil, že na 10 000 *(cca 7 000 po odfiltrování neřešitelných)* instancích už **nedojde** k výrazným odchylkám. Jak si lze všimnout, grafy se po přibližně 1000. instanci **ustálí** na hodnotách, které popisuji níže.

4.2 Porovnání metrik

Z měření metriky počet úspěšných běhů vyplývá, že zatímco gSAT úspěšně doběhl průměrně v 796 bězích, **probSAT** doběhl úspěšně v 897 bězích, probSAT byl tedy přibližně 1.1x **úspěšnější**.

Z měření metriky průměrného počtu iterací jsem zjistil, že gSAT průměrně potřeboval 611 iterací, zatímco probSATu stačilo pouhých 414. **probSAT** tedy k výsledku dospěl průměrně 1.5x **rychleji**.

Metrika váženého počtu iterací mi umožňuje porovnat, jak na tom algoritmy jsou, započteme-li také neúspěšné běhy, a to desetinásobně. gSAT po započtení penalizace potřebuje průměrně 3361 iterací, zatímco **probSATu** stačí 1801 iterací, je tedy 1.86x **rychlejší**.

Metrika **xing** umožňuje porovnat úspěšnost algoritmů pomocí posledního průsečíku korigovaných distribučních funkcí. Průměrná hodnota průsečíku je v případě, že vyhrál gSAT 916, zatímco v případě, že vyhrál probSAT 334. Znamená to, že na instancích, kde byl **probSAT efektivnější**, začal dominovat **dříve**.

Poslední metrika **winner** určuje, u kterého algoritmu ležela pravděpodobnost úspěchu výše po posledním průsečíku. Z celkového počtu 6973 instancí tato pravděpodobnost ležela 808x výše u gSATu a 6155x u probSATu, probSAT byl tedy **efektivnější 7.6x častěji**.

4.3 Závěr

Na základě porovnávaných metrik jsem došel k závěru, že probSAT jasně **dominuje** – pro každou instanci naší zadané metriky je probSAT se zadanými parametry **rychlejší nebo stejný** než gSAT se zadanými parametry.