SLOVENSKÁ TECHNICKÁ UNIVERZITA V BRATISLAVE
FAKULTA ELEKTROTECHNIKY A INFORMATIKY

MRHS SAT

TÍMOVÝ PROJEKT

2019 Alena Bednáriková, Daniel Jahodka, Fikrim Kabashi

4. MRHS SAT

Anotácia tímového projektu:

Cieľom projektu je prispôsobiť rozhrania MRHS solvera na riešenie SAT problémov vo formáte DIMACS a otestovať možnosti solvera.

Úlohy:

1. Analyzujte existujúci softvér a špecifikáciu formátov používaných SAT solvermi.
2. Navrhnite vhodné rozhranie pre integráciu SAT formátu do MRHS solvera.
3. Implementujte navrhnuté riešenie.
4. Vyhodnoťte výsledky.

Literatúra:

Zadávateľ tímového projektu:

Pavol Zajac, pavol.zajac@stuba.sk

Riešiteľský kolektív:

Alena Bednáriková, Fikrim Kabashi, Daniel Jahodka

Obsah

Ponuka

0.1 Tím

Náš tím sa skladá z 3 členov. Ide o troch programátorov, ktorí študujú na FEI STU odbor aplikovaná informatika. Všetci traja sme v minulosti spolupracovali na našich bakalárskych prácach s prof. Zajacom (zadávateľ tímového projektu) na témach úzko súvisiacich s témou tímového projektu. Jednotlivý členovia tímu:

Alena Bednáriková - programátorka. Témou jej bakalárskej práce bolo riešenie MQ problému. Jej hlavným prínosom pre tím je jej logické uvažovanie.

Daniel Jahodka - programátor. Obsahom jeho bakalárskej práce boli sústavy MRHS rovníc a ich úprava pomocou heuristických algoritmov. Jeho hlavným prínosom pre tím je riešenie problémov a návrh ich riešeni, prevažne tých programátorských.

Fikrim Kabashi - programátor so zameraním na C, C++ jazyky. Obsahom jeho bakalárskej práce bol práve SAT problém a MRHS rovnice. Implementoval algoritmus na heuristickú optimalizáciu poradia jednotlivých klauzúl v SAT formule. Práve tento algoritmus je jeden z tých, ktoré budeme implementovať do MRHS solvera.

0.2 Motivácia

Hlavnou motiváciou, prečo chceme riešiť práve tento tímový projekt je fakt, že obsah tohoto tímového projektu priamo nadväzuje na naše bakalárske práce a máme s danou témou a problematikou už nejaké skúsenosti. V našom prípade nebudeme musieť sa venovať úplným základom ohľadom tém MRHS sústav a SAT problému, ale budeme môcť sa hneď venovať novej problematike a prípadej implementácií nových algoritmov, či už na riešenie MRHS sústav/SAT problému alebo ich úprav.

0.3 Hrubý návrh a plán projektu

Nakoľko ide o vedecky zameraný projekt, tak hlavným plánom projektu je implementovať resp. doimplementovať nové algoritmy do MRHS solveru a následne vyhodnotiť dobu riešenie MRHS sústavy. Medzi plánované veci, ktoré sa budú implementovať do solvera sú:

- transformácia SAT do MRHS
- heuristická optimalizácia poradia SAT klauzúl
- SAT challenge formality
- Dual solver

0.4 Predpokladané zdroje

Najhlavnejším zdrojom, ktorý budeme potrebovať pre urýchlenie výpočtov je prístup k školskému klastru `https://www.hpc.stuba.sk`. Vďaka klastru, by sme dokázali paralelne spúšťať výpočty a úrýchliť tak získanie výsledkov a ich následné vyhodnotie.

0.5 Rozvrh

Všetci traja členovia máme rovnaký rozvrh. So zadávateľom tímového projektu máme konzultačnú hodinu dohodnutú na každu stredu semestra o 14 hodine.

0.6 Ciele riešenia

Vedúci tímového projektu prof. Pavol Zajac vyvinul softvér MRHS solver na riešenie sústav s viacerými pravými stranami. Ide o program napísaný v jazyku C. Algoritmus riešenia sústavy je popísaný v kapitole o algoritmoch. Cieľom projektu je rozšíriť tento program o nový algoritmus riešenia sústav tzv. Dual solver, o heuristickú optimalizáciu poradia klauzúl CNF formuli, o prevod CNF formuli do MRHS sústavy a o funkcionalitu v podobe špecifických výstupov a exit kódov programu, ktoré vyžaduje SAT challenge pre zúčastnenie.

Po implementovaní vyššie spomenutých algoritmov bude našou úlohou pripraviť experimenty na porovnanie dĺžky času riešenia sústavy za použitia pôvodného algoritmu MRHS solveru a za použitia Dual solver algoritmu. Vstupom algoritmov bude náhodne vygenerovaná 3-SAT (3-CNF) formula resp. 3-SAT formula upravená heuristickým algoritmom na optimalizáciu poradia klauzúl

1 Teoretické základy

V teoretickej časti tejto práce si zadefinujeme jednotlivé pojmy, s ktorými budeme ďalej pracovať. Postupne si zadefinujeme SAT problém a sústavy rovníc s viacerými pravými stranami (MRHS).

1.1 SAT

Na lepšie vysvetlenie bude potrebné zaviesť si základné pojmy z výrokovej logiky. Definovanie pojmov je spracované podľa [5] [7].

Formálna špecifikácia výrokovej logiky pozostáva z neprázdnej a neohraničenej množiny X zloženej z tzv. atomických výrokov a z množiny symbolov logických spojok $\neg$ (negácia), $\wedge$ (konjunkcia), $\vee$ (disjunkcia), $\rightarrow$ (implikácia), $\leftrightarrow$ (ekvivalencia) a pomocné symboly (zátvorky), ktoré definujú jazyk výrokovej logiky. Atomické výroky budeme označovať výrokovými premennými $(x_1, x_2, x_3, ...)$. Atomické výroky nadobúdajú pravdivostné hodnoty $x_i \in \{FALSE, TRUE\}$. Jazykom výrokovej logiky môžeme definovať tzv. výrokové formuly.

Definícia 1.1. Formula výrokovej logiky je definovaná pomocou nasledujúcich syntaktických pravidiel:

1. Každá výroková premenná $x_i \in X$ je výroková formula.
2. Ak A a B sú formule, tak $\neg A, (A \wedge B), (A \vee B), (A \rightarrow B), (A \leftrightarrow B)$ sú tiež formule.
3. Každá formula vzniká konečným použitím pravidiel (1.) a (2.).

Formuly sú splniteľné , ak dosadenie nejakých hodôt za výrokové premenné a vyhodnotením logických spojok štandartným spôsobom dostaneme pravdivostnú hodnotu $TRUE$. Napr.následovnú formulu vyhodnotíme ako splniteľnú, ak $B = TRUE$.

$$(A \vee B) \wedge (\neg A \vee B) \tag{1}$$

Pre naše potreby, si zadefinujeme ešte jeden špecialny tvar formúl výrokovej logiky tzv. konjuktívnu normálnu formu v skratke (CNF).

Definícia 1.2. [5] Výroková formula F je v konjuktívnej normálnej forme, ak má tvar konjunkcie (AND) konečného počtu klauzúl f_i, kde každá klauzula je dizjunkciou (OR) literálov (atomické výroky a ich negácie). Napríklad:

$$F = (x_1 \vee \neg x_2) \wedge (x_2 \vee x_3) \wedge (\neg x_1 \vee x_3) \tag{2}$$

F je CNF formula s troma premennými a troma klauzulami. Formulu F vyhodnotíme ako splniteľnú, ak $x_1 = x_2 = x_3 = TRUE$.

Klauzula, ktorá neobsahuje žiadny literál, sa nazýva prázdna a vyhodnocuje sa ako $FALSE$. Nastavenie logických premenných na nejaké hodnoty označíme pojmom interpretácia(formuly).

SAT je problém rozhodnutia, či existuje interpretácia, ktorá spĺňa daný booleovský výraz. Inými slovami, pýtame sa, či premenné daného booleovského výrazu, môžu byť nahradené hodnotami $TRUE$ alebo $FALSE$ takým spôsobom, že výraz sa vyhodnotí ako $TRUE$. Ak je to tak, výraz sa nazýva splniteľný. Na druhej strane, ak takéto priradenie neexistuje, výraz je vyhodnotený ako $FALSE$ pre všetky možné priradenia premenných, a výraz je nesplniteľný.

SAT problém je prvý známy NP-úplný problém (dokázané Stephenom Cookom[3]). To znamená, že všetky problémy v triede zložitosti NP sú nanajvýš tak ťažké, ako SAT. Doposiaľ, nie je známy algoritmus, ktorý efektívne rieši každý SAT problém a všeobecne sa verí, že takýto algoritmus neexistuje.

1.2 MRHS rovnice a sústavy

V tejto podkapitole si zadefinujeme pojem rovnice s viacerými pravými stranami ako aj sústavu týchto rovníc podľa článku [9]. Všetky výpočty sú uskutočnené nad poľom s dvoma prvkami (0 a 1) a označením $\mathbb{F}$. Všetky vektory $\mathbb{F}$ sú riadkové vektory a sú označené malým písmenom abecedy. Množiny sú značené veľkými písmenami abecedy a matice tučnými veľkými písmenami.

Definícia 1.3. [9] Rovnicou s viacerými pravými stranami (MRHS) nad poľom $\mathbb{F}$ nazývame výraz tvaru

$$x\mathbf{M} \in S,$$

kde $\mathbf{M}$ je matica s rozmermi $n \times l$, kde n je počet riadkov matice $\mathbf{M}$ a l je počet stĺpcov, a $S \subset \mathbb{F}^l$ je množina l-bitových vektorov. Hovoríme, že $x \in \mathbb{F}^n$ je riešením MRHS rovnice práve vtedy, keď $x\mathbf{M} \in S$.

Sústavou rovníc s viacerými pravými stranami $\mathcal{M}$ je množina m MRHS rovníc s rovnakou dimenziou n (rovnaký počet riadkov). Formálne sa môže zapísať

$$\mathcal{M} = \{x\mathbf{M}_i \in S_i | 1 \leq i \leq m\},$$

kde každé $\mathbf{M}_i$ je $(n \times l_i)$ matica a $S_i \subset \mathbb{F}^{l_i}$. Vektor $x \in \mathbb{F}^n$ je riešením sústavy MRHS rovníc $\mathcal{M}$, ak je tento vektor riešením pre všetky MRHS rovnice tejto sústavy.

Združená matica sústavy: Sústave MRHS rovníc $\mathcal{M} = \{x\mathbf{M}_i \in S_i\}$ môžeme spojiť všetky matice $\mathbf{M_i}$, nakoľko počet ich riadkov je rovnaký. Výslednú maticu označujeme $\mathbf{M}$ a nazývame ju združenou maticou sústavy:

$$\mathbf{M} = [\mathbf{M_1}|\mathbf{M_2}|\cdots|\mathbf{M_m}]$$

Takisto budeme označovať množinu vektorov pravých strán $S_1 \times S_2 \times \cdots \times S_m$ ako S. Nájdenie riešenia znamená nájdenie takého $x \in \mathbb{F}^n$, pre ktoré platí $x\mathbf{M} \in S$.

Pozn.: Stĺpce, ktoré patria $\mathbf{M_i}$ nazývame blok. V texte sa často budeme odvolávať na i-ty blok matice $\mathbf{M}$.

2 Algoritmy

V tejto kapitole si predstavíme všetky algoritmy, ktoré budeme využívať. Algoritmus na riešenie MRHS sústav, ktorý využíva MRHS solver je ukázaný v kapitole 2.1. Ostatné algoritmy sú algoritmy, ktoré sme implementovali do MRHS solvera a to prevod SAT problému na MRHS sústavu, heuristická optimalizácia poradia klauzúl v CNF formuli, Dual solver algoritmus na riešenie sústavy a transformácia MRHS sústavy na formát, ktorý vyžaduje Dual solver algoritmus na vstupe.

2.1 Algoritmus na riešenie MRHS sústavy

V tejto podkapitole si ukážeme algoritmus riešenia MRHS sústav, ktorý je implementovaný v MRHS solveri. Zápis tohoto algoritmu je prevzatý z [6] a [9]

Algoritmus 1 Uprav a vypočítaj sústavu MRHS rovníc

Vstup: MRHS sústava $x\mathbf{M} \in S = S_1 \times S_2 \times \cdots \times S_m$

Výstup: Množina $X \subset \mathbb{F}^n$, pre ktorú platí $x\mathbf{M}' \in S, \forall x \in X$, kde $\mathbf{M}'$ je upravená matica

  {ÚPRAVA ZDRUŽENEJ MATICE}

  Uprav maticu **M** na redukovaný stupňovitý tvar cez riadkové úpravy.

  Vo všetkých blokoch, ktoré obsahujú pivoty, vynuluj riadky, v ktorých sa pivoty nachádzajú, cez sĺpcové úpravy.

  {REKURZÍVNE HĽADANIE RIEŠENIA}

  $X = \emptyset$, $k = 1$, x rozšírené o bity z $S_1[1]$

Vstup: k, matica **M**, čiastočné riešenie x, množina X

  if k <= m **then** {m je počet blokov}

    for j := 1, j<= n, j++ **do** {n je počet vektorov v množine S_k}

      if $x\mathbf{M_k} == S_k[j]$ **then**

        if k == m **then**

          $X := X \cup \{x\}$

        end if

        rozšír x o bity z $S_k[j]$

        $X := X \cup \mathit{rekurzia}(k+1, \mathbf{M}, x, X)$

      end if

    end for

  end if

  return X

Tento algoritmus vráti len riešenia upravenej sústavy, a preto je nutné na základe vzorca

$$x\mathbf{A}^{-1}\mathbf{AMB} \in S_M\mathbf{B},$$

možno spätne vyjadriť riešenie pôvodnej sústavy. Ak pre upravenú sústavu platí vzťah $\mathbf{M}' = \mathbf{AM}$, potom pre riešenia upravenej sústavy platí vzťah $y = x\mathbf{A}^{-1}$. Keď osamostatníme x dostaneme vzorec $x = y\mathbf{A}$, ktorým spätne vyrátame riešenia pôvodnej sústavy [6].

2.2 Prevod SAT problému na MRHS

Každá klauzula bude u nás predstavovať jeden blok matice o veľkosti $m \times n$, kde m je počet literálov v klauzule a n je počet všetkých literálov. Počet blokov hlavnej matice, je rovný počtu klauzúl SAT problému. Prevod z formátu DIMACS na MRHS urobíme pomocou algoritmu v pseudokóde, ktorý je popísaný v algoritme 3.

Pravé strany dostaneme tak, že si vypíšeme všetky možné riešenia pre danú klauzulu, ktorých je 2^n, kde n označuje počet literálov v klauzule, a odstránime práve jedno riešenie, pre ktoré by klauzula nadobudla hodnotu $FALSE$. Z toho dostávame že počet pravých strán je $2^n - 1$.

Algoritmus 2 Vytvorenie pravých strán

Vstup: Matica **S** {Matica **S** z ALGORITMU 3}

Výstup: Pole matíc $\mathbb{P}$

Inicializuj $\mathbb{P}$ a **R**

for i := 0, i < s, i++ **do** {s je velkosť **S**}

Vytvor pravdivostnú tabuľku pre N premenných, kde N je veľkosť **S**[i] a ulož ju do **R**

{Vytvorenie zakázaného riešenia}

Vstup: Pole K {K obsahuje i-ty riadok **S**}

Výstup: Pole zakázané riešenie Z

Inicializuj pole Z

for j := 0, j < m, j++ **do** {m je veľkosť K}

if $K[j] > 0$ **then**

Vlož hodnotu 0 do Z

else

Vlož hodnotu 1 do Z

end if

end for

Nájdi Z v **R** a vymaž ho. Vlož R do $\mathbb{P}$

end for

return $\mathbb{P}$

Algoritmus 3 Vytvorenie hlavnej matice MRHS

Vstup: SAT problém vo formáte DIMACS

Výstup: Matica **S**

```
Inicializuj premenné p, k, count, pole L a maticu S
Načítaj prvý riadok do L
while L[0] != "p" do
   Načítaj ďalší riadok
end while
Vyparsuj číselné hodnoty z L a ulož ich do premennej p a k (počet premených a počet klauzúl).
while count == k do
   Inicializuj pole T
   Načítaj riadok a ulož ho do T
   Vlož T do S
   count + +
end while
return S
{Vytvorenie hlavnej matice}
```

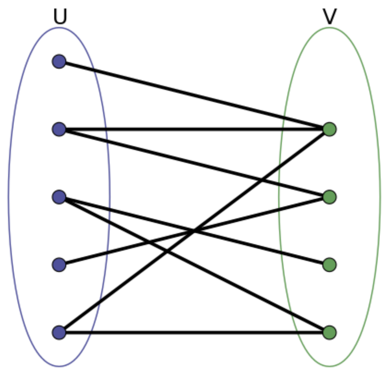
Vstup: Matica **S**, premenná p

Výstup: Pole matíc $\mathbb{B}$ {Každá matica reprezentuje jeden blok}

```
Inicializuj pole matíc 𝔹
Inicializuj maticu H a pole L
for i := 0, i < s, i + + do {s je veľkosť matice S (počet klauzúl)}
   for j := 0, j < p, j + + do {p je počet premenných}
      for k := 0, k < r, k + + do {r je veľkosť S[i] (počet literálov v klauzule)}
         if S[i][k] == j + 1 || S[i][k] == −(j + 1) then
            Vlož hodnotu 1 do L
         else
            Vlož hodnotu 0 do L
         end if
      end for
      Vlož L do H a vyčisti L
   end for
   Vlož H do 𝔹 a vyčisti H
end for
return 𝔹
```

2.3 Heuristická optimalizácia poradia klauzúl

V tejto kapitole si ukážeme heuristický algoritmus na preusporiadanie klauzúl z práce [7]. Majme bipartitný graf $G = (U, V, E)$ viď obrázok 1:



Obrázok 1: Bipartitný graf [11]

kde $|U| = n$ sú premenné, $|V| = m$ sú klauzuly, E obsahuje hranu (v_i, u_j), ak v i-tej klauzule i vystupuje j-ta premenná.

Uvažujme, že klauzuly sú v nejakom poradí P, čomu zodpovedajú číslované vrcholy $v_1, v_2, ..., v_m$. Uvažujme postupnosť indukovaných podgrafov (U^i, V^i, E^i), kde $V^i = v_1, v_2, ..., v_i$, a U^i, E^i, obsahujú všetky vrcholy a príslušné hrany spojené s V^i.

Definujme postupnosť d, kde $d_0 = 0$ a $d_i = |U^i| - |U^{i-1}|$ (čiže počet vrcholov $|U|$ pridaných do podgrafu, ak pridáme v_i).

Úlohou je nájdenie poradia P vrcholov z V také, že postupnosť d je minimálna v lexikografickom poradí (t.j. $d^P <= d^R$ ak existuje k také, že $d^P_k <= d^R_k$ a pre všetky $i < k : d^P_i = d^R_i$).

Klauzuly v algoritme č. 4 sú reprezentované ako matica, kde každý riadok matice reprezentuje jednu klauzulu. Na prvom indexe každého riadku, je číslo klauzuly a potom nasledujú premenné, ktoré obsahuje, ale iba v kladnej forme.

Premenné sú reprezentované tiež v matici, kde každý riadok matice reprezentuje jednu premennú. Na prvom indexe každého riadku, je číslo premennej a potom nasledujú čísla klauzúl, s ktorými je daná premenná spojená.

Algoritmus 4 Heuristická optimalizácia poradia klauzúl

Vstup: Matica **S** {Matica **S** z ALGORITMU 3}

Výstup: Pole P

Inicializuj maticu **V** a **U**

for i := 0, i < s, i++ **do** {s je veľkosť **S**}

Zoraď **V** od najmenšieho stupňa

Zoraď **U** od najväčšieho stupňa

Inicializuj MIN (klauzuly s minimálnym stupňom) a MAX (premenné s maximálnym stupňom)

Nájdi všetky klauzuly v **V** s minimálnym stupňom a vlož ich hodnotu na indexe 0 do MIN

if $m == 1$ **then** {m je veľkosť MIN}

Vlož $MIN[0]$ do P

else

Nájdi všetky premenné z **U** s maximálnym stupňom, ktoré sú spojené s nejakou klauzulou z **V** a vlož ich hodnotu na indexe 0 do MAX

if n == 1 **then** {n je veľkosť MAX}

Vlož hodnotu na indexe 0 z klauzule z V ktorá je spojená s premennou z **U** do P

else

Vyber náhodne jeden index z MAX a vlož index klauzule, ktorá je spojená s touto premennou do poľa P

end if

end if

Odstráň vybranú klauzulu z **V** a všetky premenné z **U** s ňou spojené.

end for

return P

Úlohou algoritmu je vyskladať novú sústavu, takým spôsobom, že každá pridaná klauzula do sústavy pridá čo najmenej nových premenných. Inak povedané, sú preferované klauzuly, ktoré sa vyskytujú v najviac klauzulách CNF formuly.

2.4 Transformácia pre Dual space solver

V tejto kapitole si ukážeme algoritmus transformácie MRHS sústavy pre algoritmus Dual solver.

Ako prvé je potrebné vyrobiť kontrolnú maticu **H** ku matici **M**. Matica **M** má rozmery $n \times l$, kde n je počet riadkov matice a l je počet stĺpcov. Po upravení matice **M** na redukovaný stupňovitý tvar, zoberieme $n - l$ posledných sĺpcov a pridáme na spodnú časť matice maticu identity o veľkosti $n \times n$. Týmto dostaneme kontrolnú maticu **H**, pre ktorú platí $\mathbf{MH} = 0$.

Ako druhé vytvoríme maticu **S**, ktorú vyrobíme nasledovne:

$$\mathbf{S} = \begin{pmatrix} S_1 & 0 & \dots & 0 \\ 0 & S_2 & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 01 & \dots & S_m \end{pmatrix}$$

Matica **S** je blokovo diagonálna matica vytvorená z množín pravých strán sústavy. Pre správnu pravú stranu s musí platiť $x\mathbf{M} = s$ a teda musí platiť aj $s \cdot \mathbf{H}^T = 0$. Výstupom algoritmu bude matica **V**, pre ktorú platí $\mathbf{V} = \mathbf{S} \cdot \mathbf{H}^T$ [13]. Táto matica bude vstupom pre Dual solver algoritmus.

Nižšie je spísaný pseudokód pre tento algoritmus.

Algoritmus 5 Vytvorenie Dual solver vstupnej matice

Vstup: MRHS sústava $x\mathbf{M} \in S = S_1 \times S_2 \times \dots \times S_m$

Výstup: Matica **V**

Uprav maticu **M** na redukovaný stupňovitý tvar.

Zober $n - l$ posledných sĺpcov a pridaj k nim maticu identity o veľkosti $n \times n$. Výslednú maticu označíme **H**

Vytvor blokovo diagonálnu maticu **S** z množiny pravých strán sústavy.

Vytvor maticu **V**, pre ktorú platí $\mathbf{V} = \mathbf{S} \cdot \mathbf{H}^T$

return **V**

2.5 Dual space solution algoritmus

Dual space solution algoritmus, narozdiel od pôvodného algoritmu 1 riešenia MRHS sústavy nie je závislý od poriadia rovníc vrámci sústavy. Princíp algoritmu je založený na subset sum probléme [12], na nájdení určitého súčtu v danej množine. V našom prípade pôjde o nájdení súčtu 0 v rámci stĺpcu matice **V**. Z toho vyplýva predpoklad, že tento algoritmus bude silno závislý na hustote 1 v matici **V** a poradí stĺpcov. Poradie stĺpcov môže byť zmenené pomocou lineárnej algebri na stĺpcoch bez zmeny pôvodného riešenia MRHS sústavy.

Bližší popis algoritmu spolu s pseudokódom môžete nájsť v článku [13], ktorý vznikol ako produkt tohoto tímového projektu. Výsledky experimentov dual space solution algoritmu nájdete v kapitole o experimentoch.

3 Dokumentácia k programu

V tejto kapitole si zhrnieme funkcionalitu, ktorá bola pridaná do MRHS solvera na základe SAT challenge špecifikácie a algoritmov.

3.1 SAT challenge špecifikácia

V tejto kapitole je stručne popísaná SAT challenge špecifikácia. Zdroj: [4]. Nakoľko tento tímový projekt je viac zameraný na vedeckú činnosť, zo SAT challenge špecifikácie sme si privlastnili len input a output formáty, ktoré ma náš program spĺňať.

3.1.1 DIMACS input formát

DIMACS CNF formát, je štandardný vstupný formát pre SAT solvery. Súbor DIMACS začína komentármi, ktoré sú na začiatku označené malým písmenom c. Prvý riadok po komentári, je v tvare $p\ cnf\ i\ j$. Premená i označuje presný počet premenných a j presný počet klauzúl obsiahnutých v súbore, potom nasledujú klauzuly. Každá klauzula môže obsahovať nenulové číslo z rozpätia $-i$ až po i. Záporné čísla označujú negáciu. Každý riadok, v ktorom je zapísaná klauzula, je ukončený nulou. Jeden riadok nemôže obsahovať navzájom opačné literály. Riadok, ktorý začína písmenom x, označuje klauzulu v XOR tvare.

```
c Súbor začína komentármi
p cnf 5 3
1 -5 4 0
-1 5 3 4 0
x 3 4 0
```

Obrázok 2: Príklad DIMACS vstupného súboru.

V logickom tvare by vyzerala formula z obrázka č. 3 následovne:

$$(x_1 \vee \neg x_5 \vee x_4) \wedge (\neg x_1 \vee x_5 \vee x_3 \vee x_4) \wedge (\neg x_3 \vee \neg x_4) \wedge (x_3 \vee x_4)$$

3.1.2 Output formát

Do výstupu programu boli pridané vypísi splniteľnosti vstupnej k-SAT klauzule. V nižšie zobrazenej tabuľke sú zapísané exit kódy programu spolu s vypísom splniteľnosti, ktorý predstavujú.

Tabuľka 1: Exit kódy

exit(kód)	SAT ?
exit(10)	SATISFIABLE
exit(20)	UNSATISFIABLE
exit(0)	UNKNOWN

3.2 MRHS solver

Na riešenie sústav s viacerými pravými stranami budeme používať program MRHS solver, ktorý vyvinul prof. Ing. Pavol Zajac, phD. Algoritmus 1, ktorý sme si predstavili v kapitole 2 sa využíva práve v tomto programe na riešenie sústavy.

Jedným z naších cielov bolo implementovať aby vstupom algoritmu bol súbor formátu DIMACS ukázaný v kapitole 3.1.1. Následne aby sme mohli využívať optimalizačný algoritmus z kapitoly 2.3, museli sme rozšíriť MRHS solver o nové interface-i aby vedel pracovať s CNF štruktúrami. Nakoľko pôvodný algoritmus 1 riešenia sústavy stále pracuje so štruktúrami podobnými s tými spomenutými v nasledujúcej kapitole o vstupnom formáte súborov pre MRHS solver, bolo nutné implementovať algoritmus z kapitoly 2.2 aby sme k-SAT problém transformovali do MRHS sústavy. Následne takto transformovaný k-SAT problém je možné riešiť MRHS solverom.

V prípade, že máme MRHS sústavu v takto pôvodnom tvare, je možné pomocou algoritmu 5 transformovať sústavu do matice **V**, ktorú vie vypočítať algoritmus Dual space solver spomenutý v kapitole 2.5 a podrobnejšie spísaný v článku [13].

3.3 Nová funkcionalita

Nakoľko MRHS solver je konzolová aplikácia, ktorej účelom je vypočítať sústavu rovníc s viacerými pravými stranami, tak celý beh programu je nastavený pri spustení programu pomocou prepínačov. Do programu boli pridané nasledovné prepínače:

- -t = počet sekúnd, po ktorých sa má program vypnúť a vrátiť exit kód UNKNOWN (0)
- -i = s hodnotou 1 zapne optimalizáciu vstupnej k-SAT formule (algoritmus 4), defaultne je zadaná hodnota nula (bez optimalizácie)
- -a = typ solvera. Pre hodnotu 0 je algoritmus riešenia sústavy nastavený na pôvodný algoritmus riešenia (alg. 1). Pre hodnotu 2 je použitý dual space solution algoritmus (alg. 2.5). Pre hodnotu 3 sa sústava vypočíta oboma algoritmami.

Prepínače programu sú stavané na jednoduchú prácu so solverom počas experimentov.

4 Experimentálna časť

Naším cieľom bolo systematicky porovnať dobu riešenia náhodne generovanej sústavy k-SAT klauzule s dobou riešenia tejto sústavy po úprave algoritmom 4. Na riešenie sústav sme použili MRHS solver s algoritmami na riešenie sústavy 1 a dual space solution z kapitoly 2.5. Na generovanie k-SAT formúl sme využili program Cnfgen [8].

4.1 Testovacia platforma

Všetky výpočty boli robené na jednom zariadení pre zaistenie čo najmenších odchýlok. Parametre počítača, na ktorom boli uskutočnené výpočty, sú:

- Procesor: Intel core i5 - 8600K, 3.6GHz
- RAM: 16GB

Výpočty prebiehali na jednom jadre procesora.

4.2 Metodika experimentov

Skúmali sme náhodné k-Sat problémy. Pre tieto testy sme si zvolili $k = 3, 4$ a menili sme rôzne faktory a sledovali ako vplývajú na výstup. Sledovanou premennou bude čas [s], za ktorý dokážu algoritmy na riešenie sústav nájsť všetky riešenia problému. Pre každú sadu testov, sa vygenerovali problémy s následovnými faktormi:

Fixný pomer m/n

Tento pomer, sme volili rôzny pre jednotlivé k podľa štúdií [1][10], tak aby sme boli v oblasti tzv. najťažších problémov.

Pevne dané n a premenlivé m

Pri tomto teste sme si nastavili spodnú hranicu tak, aby bola pravdepodobnosť splniteľnosti cca. 90% a hornú hranicu sme nastavili tak, aby bola pravdepodobnosť splniteľnosti 0% podľa vzťahu $m/n = 2^k \log 2 - \frac{\log 2}{2}$ (vzťah prebratý z [2]).
Následne sme na tieto problémy aplikovali naše algoritmy samostatne, ale aj v kombinácii. Aplikovali sa naslednovné algoritmy:

Experimenty s pôvodným algoritmom riešenia sústavy

Vygenerované k-SAT problémy boli transformované na MRHS sústavu a následne riešené algoritmom 1.

Experimenty s dual space solution algoritmom Vygenerované k-SAT problémy po transformácií na MRHS sústavu boli transformované na novú reprezentáciu problému spomenutú v kapitole 2.4 a následne riešenie pomocou algoritmu z kapitoly 2.5.

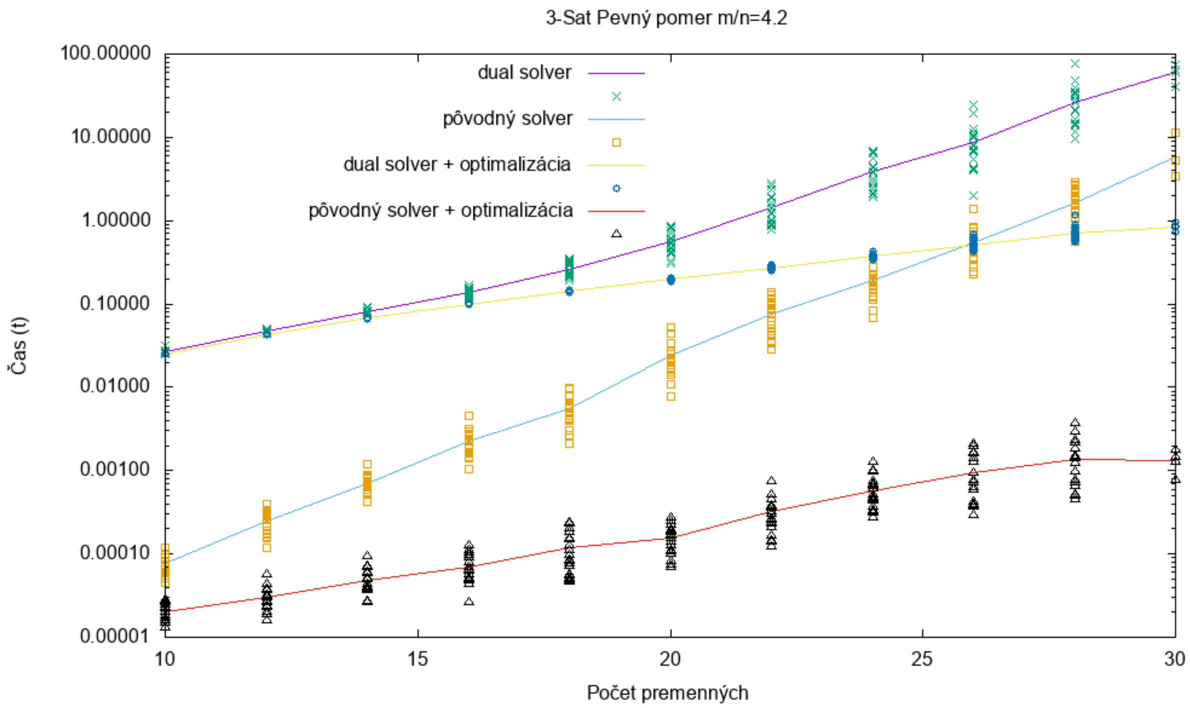
Experimenty s heuristickou optimalizáciou Na vygenerované k-sat problémy, sme aplikovali nami implementovanú heuristickú optimalizáciu poradia klauzúl. Následne sme porovnávali časy, potrebné na nájdenie všetkých riešení s využitím pôvodného algoritmu riešenia sústavy a dual space solution algoritmom.

4.3 Vyhodnotenie experimentov

Všetky grafy v tejto kapitole sú zobrazené v logarimickej mierke na y-osi.

4.3.1 3-SAT pevný pomer m/n

V tomto experimente sme skúmali vplyv zvyšujúceho sa počtu premenných n v 3-sat formule pri pomere $m/n = 4.2$.

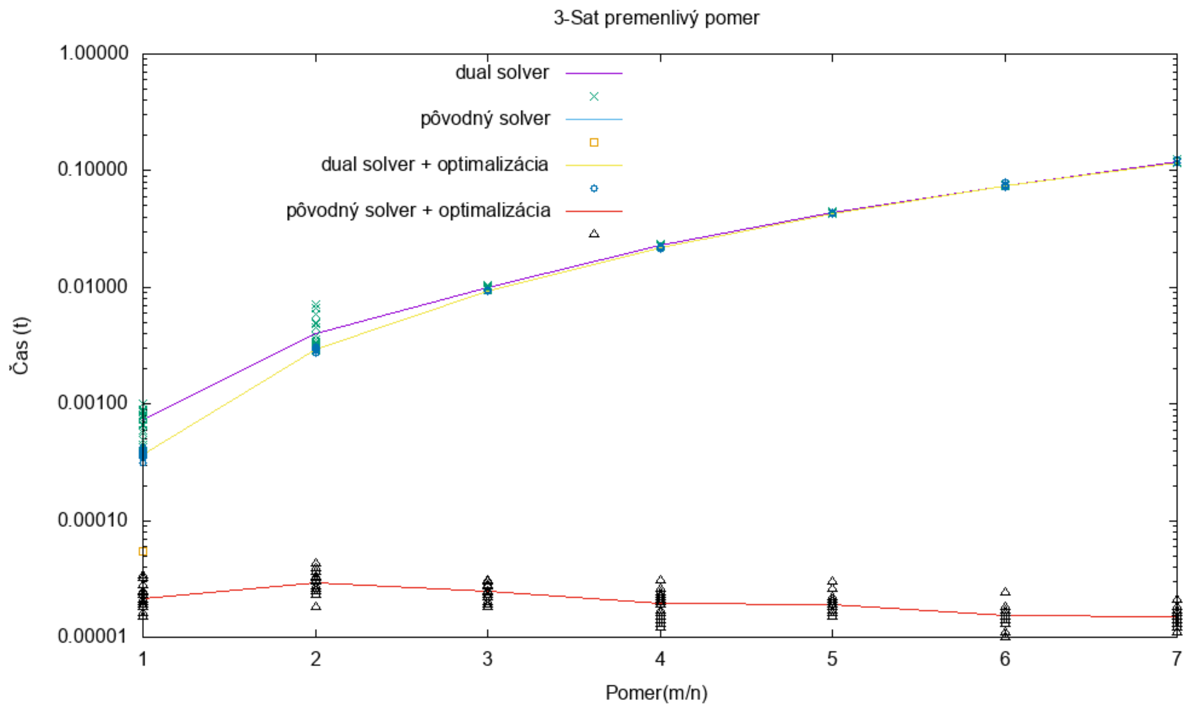


Obrázok 3: Experiment 1: 3-SAT pevný pomer m/n

Z vyššie zobrazeného obrázku 3 možno vidieť, že dual space solution algoritmus bez pre-optimalizácie exponenciálne rýchlejšie rastie v porovnaní s ostatnými spôsobmi riešenia. Najlepší čas sme dosiahli v prípade použitia pôvodného algoritmu a pre-optimalizácie 3-sat formule.

4.3.2 3-SAT premenlivý pomer

V tomto experimente sme skúmali vplyv zvyšujúceho sa pomeru m/n na dobu riešenia sústavy.

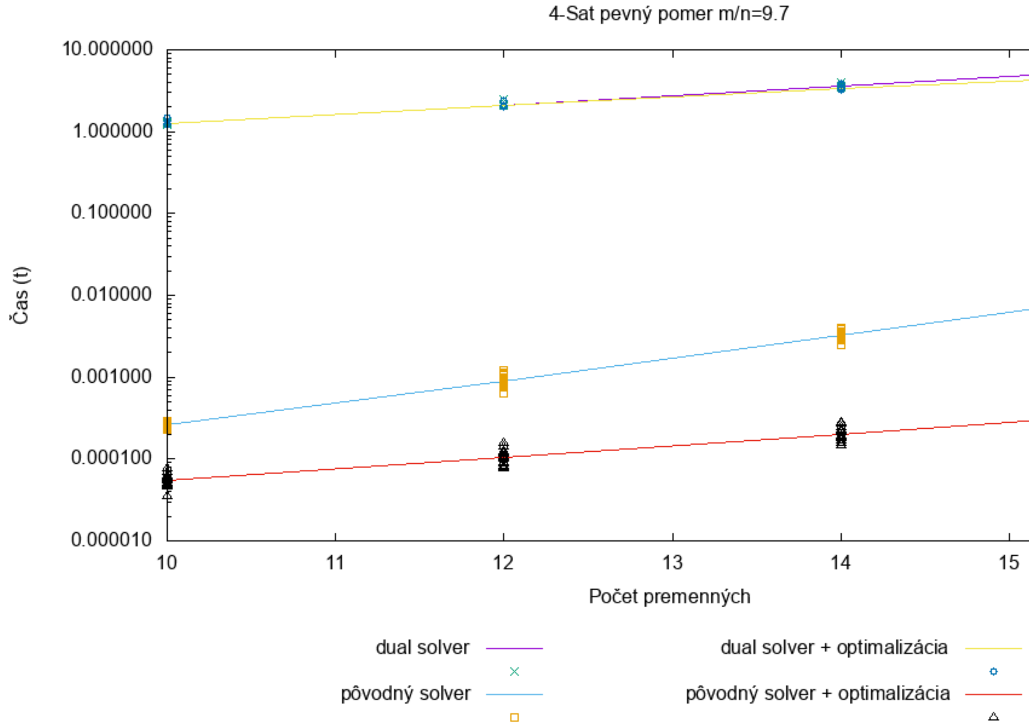


Obrázok 4: Experiment 2: 3-SAT premenlivý pomer

V prípade zvyšujúceho sa pomeru m/n možno zvyššie zobrazeného grafu vidieť, že pôvodný algoritmus riešenia s heuristickou optimalizáciou klauzúl sa od pomeru 2 začal zlepšovať. Ostatné algoritmy exponenciálne rástli so zvyšujúcim sa pomerom.

4.3.3 4-SAT pevný pomer m/n

V tomto experimente sme skúmali vplyv zvyšujúceho sa počtu premenných n v 4-sat formule pri pomere $m/n = 9.7$.

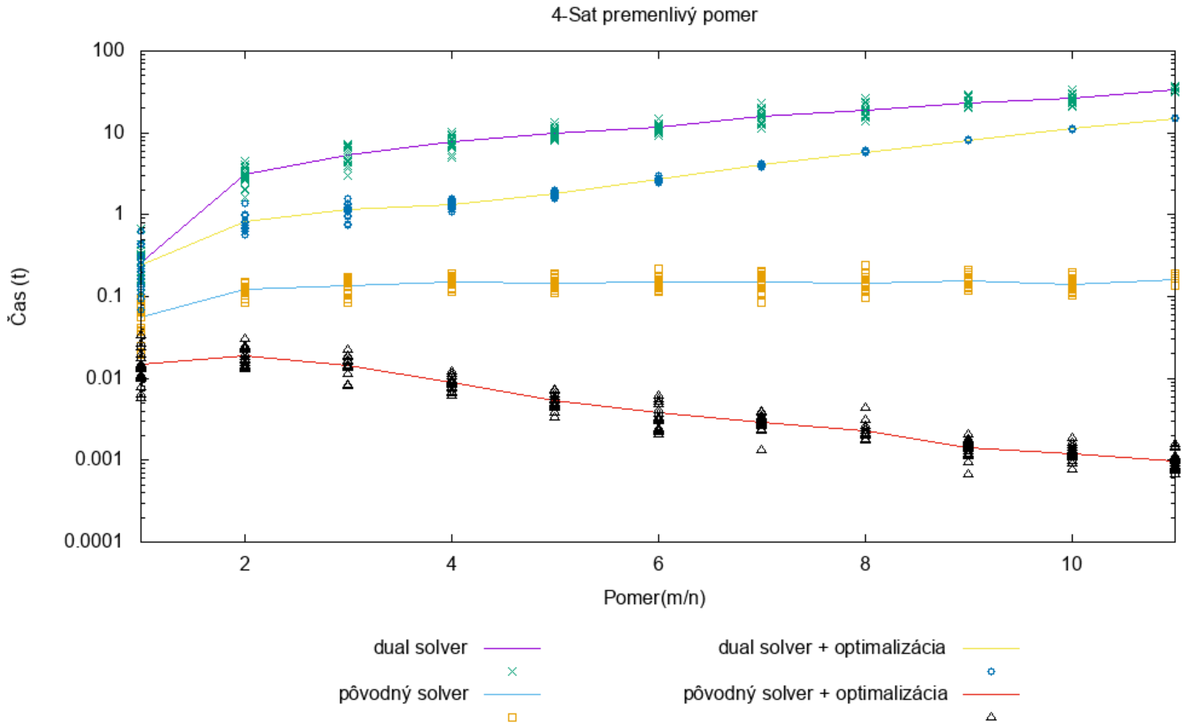


Obrázok 5: Experiment 3: 4-SAT pevný pomer m/n

Z obrázka možno jasne vidieť prevahu v čase riešenia medzi pôvodným algoritom riešenia sústavy bez použitia optimalizácie a s použitím optimalizácie. V prípade použitia optimalizácie je zrýchlenie nájdenia riešenia niekoľko násobné. Z tohoto experimentu sa dá utvrdiť predpoklad, že dual space solution algoritmus je závislý od lineárnej algebry a nie od poradia rovníc sústavy. Z grafu možno vidieť, že rozdiel medzi optimalizovanou sústavou a sústavou bez optimalizácie je nepatrný.

4.3.4 4-SAT premenlivý pomer

V tomto experimente sme skúmali vplyv zvyšujúceho sa pomeru m/n na dobu riešenia sústavy.



Obrázok 6: Experiment 4: 4-SAT premenlivý pomer

Z grafu možno vidieť, že doba riešenia sústavy za použitia pôvodného algoritmu riešenia sústavy spolu s optimalizáciou sa v prípade, zvyšujúceho pomeru zlepšuje. V prípade nepoužitia optimalizácie sa zdá byť doba riešenia sústavy s pôvodným algoritmom riešenia konštantná.

Heuristická optimalizácia dokáže niekoľko násobne zlepšiť dobu riešenia sústavy v prípade použitia dual space solution algoritmu, ale v porovnaní s pôvodným algoritmom je stále priestor na zlepšenie.

Zoznam použitej literatúry

[1] CHERTKOV, E. Phase transitions in random satisfiability problems. http://guava.physics.uiuc.edu/~nigel/courses/563/Essays_2017/PDF/chertkov.pdf, 2017.

[2] COJA-OGHLAN, A. Chasing the k-sat threshold. https://www.math.uni-frankfurt.de/~acoghlan/talk_AtlantaSAT.pdf, 2017.

[3] COOK, S. A. *The Complexity of Theorem-Proving Procedures*. Proceedings of the Third Annual ACM Symposium on Theory of Computing, 1971.

[4] HEULE, M., JARVISALO, M., AND SUDA, M. Sat competition. http://sat2018.forsyte.tuwien.ac.at/.

[5] HURTH, M., R. M. *Logic in computer science*, 2 ed. Cambridge University Press, New York, 2004. ISBN 978-0-511-26401-6.

[6] JAHODKA, D. *Heuristické riešenie MRHS rovníc*. 2018.

[7] KABASHI, F. *Aplikácia MRHS rovníc na riešenie SAT problému*. 2018.

[8] LAURIA, M. Cnfgen. https://github.com/MassimoLauria/cnfgen.

[9] RADDUM, H., AND ZAJAC, P. Mrhs solver based on linear algebra and exhaustive search. https://eprint.iacr.org/2018/111.pdf.

[10] STEPHAN MERTENS, MARCK MÉZARD, R. Z. Threshold values of random k-sat from the cavity method. https://arxiv.org/pdf/cs/0309020.pdf, 2017.

[11] WIKIPEDIA. Bipartitný graf. https://en.wikipedia.org/wiki/Bipartite_graph.

[12] WIKIPEDIA. Subset sum problem. https://en.wikipedia.org/wiki/Subset_sum_problem.

[13] ZAJAC, P., BEDNÁRIKOVÁ, A., JAHODKA, D., AND KABASHI, F. Solving mrhs systems generated from k-sat instances.