Софиийски университет "Св. Климент Охридски"
Факултет по математика и информатика

Домашна работа 1

по

Изкуствен Интелект

спец. Информатика, 3 курс, летен семестър,

учебна година 2018/19

Тема Удовлетворяване на ограниченията върху криптограма на умножение

15 април 2019 г.

София

Изготвил:

Иво Алексеев Стратев

Фак. номер: 45342

Група: 3

Съдържание

1 Задача

Решаваме Криптограма на умножение чрез алгоритъм за удовлетворяване на ограниченията. Реализирания алгоритъм може да решава произволна криптограма от зададения вид, както и произволна задача за удовлетворяване на ограниченията при задаването на подходящи за реализацията входни данни.

1.1 Търсим решение на следната криптограма

$$\begin{aligned} ABC * DEB = \\ ABC \\ +IAG0 \\ +EHFA00 = \\ EDBDFC \end{aligned}$$

Тъйкато се иска решение на конкретната криптограма ще използавме именно нея за пример в описанието на предложения/изполвания метод за решаване на задачата.

2 Описание на предложения/използвания метод за решаване на задачата

2.1 Инициализация на използваните дани

2.1.1 Намиране на ограниченията от алгоритъма

За решването на криптограмата използваме три вида ограничения:

- Ограничение All Diff (всички промени са с различни стойности) наложено от формата на решаваната задача. За конкретната задача това са 36 ограничения.
- Ограничения по умножение. За конкретната задача това са следните ограничения:

$$\begin{aligned} ABC * DEB = EDBDFC \\ B * ABC = ABC \\ B * BC \equiv BC \pmod{100} \\ B * C \equiv C \pmod{10} \\ E * ABC = IAG \\ E * BC \equiv AG \pmod{100} \\ E * C \equiv G \pmod{10} \\ D * ABC = EHFA \\ D * ABC \equiv HFA \pmod{1000} \\ D * BC \equiv FA \pmod{100} \\ D * C \equiv A \pmod{10} \end{aligned}$$

- Ограничения по сума. За конкретната задача това са следните ограничения:

$$
\begin{aligned}
000ABC + 00IAG0 + EHFA00 &= EDBDFC \\
00ABC + 0IAG0 + HFA00 &\equiv DBDFC \pmod{100000} \\
0ABC + IAG0 + FA00 &\equiv BDFC \pmod{10000} \\
ABC + AG0 + A00 &\equiv DFC \pmod{1000} \\
BC + G0 + 00 &\equiv FC \pmod{100}
\end{aligned}
$$

2.1.2 Пресмятане на статичните рангове използвани от алгоритъма при избиране на най-добра променлива за инстанциране

Когато всички ограничения са в сила и алгоритъма няма какво повече избира най-добрата от неизбраните досега променливи, на която да даде стойност - някоя стойност от текущия домейн на променливата. Най-добра променлива е някоя променлива, която е минимална относно големината на домейна си и наредбата на статичните рангове. За решаването на конкретната задача за статични рангове се използват наредени тройки от цели числа с наредба - лексикографската наредба над $\mathbb{Z}^3$. Като променлива x има статичен ранг $(count, rang, index)$. Където

- $count$ е общия брой на променливи в условията на криптограмата, от който е изваден броят на срещанията на променливата x в условията на криптограмата. Тоест $count$ е броят на срещания на променлива в условията, която не е x. Това се налага за да може променлива, която се среща повече на брой пъти (по-ограничаваща е) да има по-малък статичен ранг.

- $rang$ е число от множеството $\{1, 2, 3, 4\}$. Като $rang$ е 1, ако първото срещане на променливата е в числото, което бива умножавано. В случая това е числото ABC и следователно променливите A, B и C имат стойност 1 във втората компонента на своя статичен ранг. $rang$ е 2, ако първото срещане на променливата е в числото, с което се умножава. В случая това е числото DEB и следователно променливите D и E имат стойност 2 във втората компонента на своя статичен ранг. $rang$ е 3, ако първото срещане на променливата е в някое от числата, които са междиден резултат. В случая това са числата ABC, IAG и $EHFA$ и следователно променливите I, G, H и F имат стойност 3 във втората компонента на своя статичен ранг. $rang$ е 4, ако първото срещане на променливата е в резултата на умножението. В случая това е числото $EDBDFC$ и следователно нито една променлива няма стойност 4 във втората компонента на своя статичен ранг, понеже всяка променлива се среща по-нагоре. Конкретно за задачата $rang$ има евристична стойност, която е колко е по-малко число, ако империчната вероятност да стигнем по-бързо да резултат е по-голяма. За да достигнем до резултат дефакто имаме нужда само от стойностите на променливите, които реализират умножението (в случая това са променливите от ABC и DEB), за това и ако променлива първо биде срещната в числата, които се умножават, то тя има по-малка стойност.

- $index$ е позицията от дясно на ляво в първото числото, в което променливата участва.

Статичните рангове в задачата изчислени от алгоритъма са следните:

$$
\begin{aligned}
A &: (18, 1, 3) \\
B &: (18, 1, 2) \\
C &: (19, 1, 1) \\
D &: (19, 2, 3) \\
E &: (19, 2, 2) \\
F &: (20, 3, 2) \\
I &: (21, 3, 3) \\
G &: (21, 3, 1) \\
H &: (21, 3, 3)
\end{aligned}
$$

2.1.3 Определяне на началните домейни

По подразбиране всяка променлива в началото има домейн множеството $\{1, 2, 3, 4, 5, 6, 7, 8, 9\}$. Но се прави опит за намаляване на домейните.

- Първо се проверя най-дясната променлива от числото, което бива умножавано. В случая това е променливата B от числото ABC. Ако тя отговаря на условието: когато бъде умножена с някоя цифра на числото, с което се умножава и последната цифра на резултата е същата цифра, то домейна на тази променлива се намалява до $\{1, 3, 5, 6, 7, 9\}$. Тоест могат да бъдат отстранени три стойности. За конкретната задача това не е възможно понеже $E*C \equiv G \pmod{10}$ и $C \neq G$.
  Доказателсво на твърдението, че домейна може да бъде стеснен
  Нека променливата X е най-дясната и за някоя променлива Y от числото, с което бива умножавано е вярно, че $Y * X \equiv Y \pmod{10}$. Да разгледеме какви са възможностите за X спрямо стойността на Y:

$$
\begin{aligned}
1 * X \equiv 1 \pmod{10} &\implies X \in \{1\} \\
2 * X \equiv 2 \pmod{10} &\implies X \in \{1, 6\} \\
3 * X \equiv 3 \pmod{10} &\implies X \in \{1\} \\
4 * X \equiv 4 \pmod{10} &\implies X \in \{1, 6\} \\
5 * X \equiv 5 \pmod{10} &\implies X \in \{1, 3, 5, 7, 9\} \\
6 * X \equiv 6 \pmod{10} &\implies X \in \{1, 6\} \\
7 * X \equiv 7 \pmod{10} &\implies X \in \{1\} \\
8 * X \equiv 8 \pmod{10} &\implies X \in \{1, 6\} \\
9 * X \equiv 9 \pmod{10} &\implies X \in \{1\}
\end{aligned}
$$

  Следователно $X \in \{1\} \cup \{1, 6\} \cup \{1, 3, 5, 7, 9\} = \{1, 3, 5, 6, 7, 9\} \quad \square$.

- Проверява се дали има променлива от числото, с което се умножава и резултат от умножението с числото, което бива умножавано е същото число, то тази променлива със сигурност има стойност 1. В случая това е променливата B, защото $B * ABC = ABC$. Това е така, защото единствено числото 1 има това свойство. Ако бъде намерена единица, то се проверява дали има променлива от числото, с което се умножава, такава, че резултата от умножението на тази променлива с числото, което бива умножавано да е число със същия брой цифри (променливи) и проверявата променлива и първата променлива от числото, което е умножавано не са единицата, то домейните на проверяваната променлива и на най-лявата променлива от резултата са множеството $\{2, 3, 4\}$.
  Доказателсво на твърдението, че домейна може да бъде стеснен
  Нека $X \neq 1 \ \wedge \ S \neq 1$ и $X * STU = WYZ$ тогава, ако $X > 5 \ \vee \ S > 5$, то $X * S \geq 10$, което е абсурд понеже $\left\lfloor \dfrac{X * S}{10} \right\rfloor = W < 10$. Следователно $X \in \{2, 3, 4\} \ \wedge \ S \in \{2, 3, 4\}$. $\quad \square$
  В конкретната задача понеже $B = 1$ и $E * ABC = IAG$, то домейните на A и на E биват редуцирани.

2.2 Намиране на всички решения на задачата

. Алгоритъма по намиране на всички решения се състой от три етапа, които са независими от текущото състояние на решаваната задача (конкретната оценка на променливите). След инициализацията се преминава към Фаза 1. на алгоритъма за намиране на решения с празна оценка.

2.2.1 Фаза 1. Проверка на единичните ограничения

Единично ограничение е всяко ограничение, за което само единствена променлива участваща в ограничението няма стойност в текущата оценка. Проверката на единичните ограничения е директна.

За всяко единично ограничение c върху променлива x се премахват всички стойности от домейна на x, които нарушават условеито на ограниченито c. Ако домейна на променливата x стане празен, то текущата задача няма решение. Ако задачата продължава да има решение след удовлетворяването на единичните ограничения те биват премахнати и се преминава към Фаза 2.

2.2.2 Фаза 2. Проверка на двойните ограничения

Двойно ограничение е всяко ограничение, за което точно две променливите участваща в ограничението нямат стойност в текущата оценка. За двойните ограничения се строй граф на ограниченията върху, който се изпълнява алгоритъм за пълна съвместимост на дъгите.

- сформира се опашка от наредени тройки $variableI, variableJ, constraint$ от променлива, променлива, ограничние, по следния начин за всяка променлива $variableI$ и всяко ограничение $constraint$ върху $variableI$ и $variableJ$ към опашката се добавя $variableI, variableJ, constraint$.
- Докато опашката не е празна:
  - Взима се и се премахва първата тройка $variableI, variableJ, constraint$ от върха на опашката.
  - Редуцира се домейна на променлива $variableI$ като се премахва всяка стойност, за която няма стойност на домейна на $variableJ$, за която ограничението $constraint$ да е изпълнено.
  - Прави се проверка дали домейна на $variableI$, ако е алгоритъма по удовлетворяване на дъгите спира и връща съобщава за това.
  - В края на опашката се добавят тройки $variableK, variableI, constraintKI$ за всяко ограничение $constraintKI$, което ограничава $variableK$ и $variableI$ и $variableK$ е различна от $variableJ$.
- Връщат се редуцираните домейни на променливите.

Преминава се към Фаза 3.

2.2.3 Фаза 3. Избор на най-добра променлива за инстанциране и премиване към решаването на подзадачи

Променлива, която се инстанцира е таква, на която се избира да бъде даде стойност.

- Избира се най-добра променлива за инстанциране - променлива, която има най-малко елементи в домейна си и е с най-нисък статичен ранг.
- От информацията за всяко ограничение се премахва информацията, че ограничението служи за определяне на стойността на избраната променлива.
- Решението на текущата задача е обединието на решенията на под задачите. За всяка стойност на домейна на избраната променлива се сформира подзадача, като към оценката на текущата задача се добавя информацията, че избраната променлива има стойност - съответната стойност от домейна и тази задача се решава, като се преминава към Фаза 1.

3 Описание на реализацията с псевдокод

3.1 Псевдокод на алгоритъма за решаване на задача удовлетворяване на ограниченията

function SolveCSP(constraintsWithVariablesInfo, rangs, domains)
  return Solutions(constraintsWithVariablesInfo, rangs, domains, {})
end function

function Solutions(csWithV, rs, ds, e)
  $sCs \leftarrow$ SelectConstraints($csWithV, 1$)
  for all $(c, var) \in sCs$ **do**
    $dom \leftarrow \{val \in ds[var] \mid \textsc{Check}(c, e\langle(var, val)\rangle)\}$
    if $dom = \emptyset$ **then**
      return {}
    else
      $ds[var] \leftarrow dom$
    end if
  end for
  $csWithV \leftarrow csWithV \setminus sCs$
  $arcCs \leftarrow$ SelectConstraints($csWithV, 2$)
  $vs \leftarrow \{var \mid (var, rang) \in rs\}$
  $mDs \leftarrow$ ArcConsistancy($vs, arcCs, ds, e$)
  if $mDs = Nothing$ **then**
    return {}
  end if
  $ds \leftarrow$ fromJust(mDs)
  $bv \leftarrow$ BestVariable(ds, rs)
  $rs \leftarrow \{(v, r) \in rs \mid v \neq bv\}$
  $csWithV \leftarrow \{(c, vars \setminus \{bv\}) \mid (c, vars) \in csWithV\}$
  return $\bigcup_{val \in ds[bv]} \textsc{Solutions}(csWithV, rs, ds, e\langle(bv, val)\rangle)$
end function

function SelectConstraints(csWithV, count)
  return $\{(c, vs) \in csWithV \mid |vs| = count\}$
end function

3.2 Псевдокод на алгоритъма пълна съвместимост на дъгите

```
function ArcConsistancy(vs, cs, ds, e)
    q ← {}
    for all vI ∈ vs do
        for all (c, {vI, vJ}) ∈ {(c, vars) ∈ cs | ∃varJ ∈ vs : vars = {vI, vJ}} do
            q ← q ∪ {(vI, vJ, c)}
        end for
    end for
    while q ≠ {} do
        (vI, vJ, c) ← Enqueue(q)
        dom ← {valI ∈ ds[vI] | ∃valJ ∈ ds[vJ] : Check(c, e⟨(vI, valI), (vJ, valJ)⟩)}
        if dom = {} then
            return Nothing
        else
            ds[vI] ← dom
            for all (cK, {vK, vI}) ∈ aCs do
                if vK ≠ vJ then
                    q ← q ∪ {(vK, vI, cK)}
                end if
            end for
        end if
    end while
    return Just ds
end function
```

За краткост за модифцирне на оценката са използвани $\langle\ \rangle$.

4 Инструкции за компилиране на програмата

Трябва да имате инсталиран GHC (Glasgow Haskell Compiler) на системата. Най-лесно това става чрез инструмента Stack https://docs.haskellstack.org/en/stable/README/. Следвайте инструкциите за инсталиране на сайта. Компилиране и изпълнение на програмата:

$$\$ \quad stack \quad ghc \quad ./Main.hs$$
$$\$ \quad ./Main$$

5 Резултати

При изпълнение на програмата се намира единсвено решение на поставената задача

$$[(B, 1), (A, 3), (E, 2), (C, 7), (D, 9), (F, 5), (G, 4), (I, 6), (H, 8)]$$

Проверката, че решението е единствено става посредством втора програма, която генерира всички пермутации от възможни стойностии (Пермутации заради ограничението All Diff) и принтира всяко едно решение (пермутация/остойностяване на променливите). Компилиране и изпълнение на програмата за генериране и тестване:

$$\$ \quad stack \quad ghc \quad ./tester.hs$$
$$\$ \quad ./tester$$

Резултат:

$$[('A', 3), ('B', 1), ('C', 7), ('D', 9), ('E', 2), ('F', 5), ('I', 6), ('G', 4), ('H', 8)]$$