Изкуствен интелект
Домашно №1

Ростислав Стоянов
ф-н 45244

1 Описание на използвания метод за решаване на задачата

За целите на решаване на поставената задача същата се разглежда като задача от вида "търсене в пространството на състоянията и по-конкретно от вида задачи, свързани с намирането на цел при спазване на ограничителни условия.Дефинираме допълнителните променливи T,U...Z и разглеждаме следните ограничения:

- EC = G + 10T
- T + E = A + 10U
- U + EA = I
- DC = A + 10V
- D + V = F + 10 W
- W + AD = H + 10 E
- B+ G = F + 10 X
- X +3A = F + 10 Y
- Y+ I + F = B + 10 Z
- Z + H = D

За решаването на задачата се използва backtracking алгоритъм (търсене с възврат) заедно с евристика. Използваната евристика е избор на най - ограничена променлива, т.е. всеки път, когато избираме променлива, на която да присвоим стойност се избира тази, чийто домейн е най-малък, а при равенство се избира тази променлива, която е свързана с най-много други променливи чрез ограниченията. Целта на тази евристика е да "сгрешим рано"и да намалим цената на възврата. От ограниченията и условието на задачата лесно се вижда, че B = Z =1 ,което се използва за начално състояние.

2 Псевдокод и описание на реализацията

Следва псевдокод на основните части от решението на задачата.

```
1: procedure BACKTRACK (STATE currState)
2:     if !canContinue(currState) then
3:         return (isFinal(currState))
4:     ans ← false
5:     varChoice ← smallestDomain(state)
6:     domain ← domains.varChoice
7:     for digit in domain do
8:         if digit is used then
9:             continue
10:        newState ← currState + var ← digit
11:        if consistentWithConstraints(newState) & backtrack(newState) == true then
12:            ans ← true
13:        destroy newState
14:    return ans
```

Това е основната част от решението на задачата - тук се реализира търсенето с възврат. Функцията canContinue проверява дали на всяка променлива е присвоена стойност, а функцията isFinal проверява дали получените стойности удовлетворяват условията и ако да, печата полученото решение на стандартния изход.

```
1: procedure SMALLESTDOMAIN (STATE currState)
2:     size ← ∞
3:     var ← 'L'
4:     list ← {}
5:     for domain in currState.domains do
6:         currSize ← domain.size()
7:         if currSize < size & currSize > 1 & var ϵ {A,..,E} then
8:             size ← currSize
9:             var ← domain.var
10:            list ← {}
11:        if currSize == size & currSize > 1 & var ϵ {A,..,E} then
12:            list ← {list,domain.var}
13:        if currSize == 'L' then
14:            var ← getMostConstrainingValue(list)
15:        if currSize == 'L' then
16:            return getFirstPossible
17:    return var
```

Функцията smallestDomain връща променливата, чийто домейн е най-малък в момента. Ако има повече от един такъв домейн, и има променлива от началните, се връща най - ограничаващата променлива. За целта се използва функцията getMostConstrainingValue, която приема лист от променливи, който да разглежда и връща тази,която е във връз-

ка с най-много други променливи, използвайки графа на ограниченията. Ако такава липсва, се връща първата възможна променлива. Когато се създава ново състояние, и се задава стойност на някоя от променливите, която няма стойност, се променят домейните на всяка една от променливите като за целта се използват въведените от преди това ограничения.

3 Инструкции за компилация

Компилацията може да се извърши по два начина като за целта е необходим C++ компилатор поддържащ стандарта C++ 14 (и cmake версия>=3.14 ако се използва) Първият начин е да се компилират отделно всички класове и после да се свържат с main file-a. Това може да стане (използвайки g++) като се изпълният командите :
g++ -std=c++1y -c Assignment.cpp,
g++ -std=c++1y -c State.cpp,
g++ -std=c++1y -c Constraints.cpp,
g++ -std=c++1y -c Domain.cpp,
g++ -std=c++1y Assignment.o State.o Constraints.o Domains.o main.cpp .
Полученият изходен файл може да се изпълни от терминал. Алтернативно може да се използва CMake като за целта в архива се съдържа CMakeLists.txt файл, с чиято помощ да се компилира програмата. Това може да стане като се изпълнят следните(примерни) команди:
mkdir build
cd build
cmake ..
make
Резултат от изпълнението на командите е изпълнимият файл hw1.
Забележка: Използваните команди са примерни - в зависимост от използваната операционна система може да има разлика при компилацията

4 Резултати

Описаното решение на задачата намира 1 решение на поставения проблем. Това е решението A = 3, B = 1, C = 7, D = 9, E = 2, F = 5, G = 4, H = 8, I = 6; T = 1, U = 0, V = 6, W = 1, X = 0, Y = 0, Z = 1. Проверка за коректност на решението се прави преди то да бъде изведено на стандартния изход т.е. гарантира се, че $ABC \times DEB = 317 \times 921 = 291957 = EDBDFC$