Verificare Formala, WS2023/2024. Lab 2

Se incarca un fisier pdf cu rezolvarile. Se incarca sursele programelor si outputul acestora. Aditional outputul se include si ca screenshot in fisierul pdf.

1 Plasare maşini virtuale pe servere

Formalizati si rezolvati folosind Z3 urmatoarea problema (in toate cele 4 variante). Presupunem ca avem 3 masini virtuale care necesita 100, 50 si, respectiv, 15 GB memorie hardware. Exista 3 servere cu proprietatile 100, 75 si 200 GB memorie hardware in aceasta ordine. Gasiti o modalitate de a plasa masinile virtuale pe servere pentru a:

1. Minimiza costul operational (serverele au costuri zilnice fixate astfel: 10, 5 si, respectiv, 20 USD)
2. Minimiza numarul de servere folosite.

Indicatii: Mai multe detalii se afla in fisierul Cursul4.pdf incarcat o data cu laboratorul.
Notam cu x_{ij} faptul ca masina virtuala i se poate aloca serverului j si cu y_j denotam ca serverul j este dat in folosinta. Trebuie sa specificam urmatoarele:

1. O masina virtuala se aloca doar unui server:

$$x_{i1} + x_{i2} + x_{i3} = 1, \quad \forall i \in \{1, ..., 3\}$$

2. Un server este activ daca gazduieste o masina virtuala:

$$(x_{1j} = 1) \vee (x_{2j} = 1) \vee (x_{3j} = 1) \Rightarrow (y_j = 1), \quad \forall j \in \{1, ..., 3\}$$

3. Constrangeri de capabilitate/hardware

$$100x_{11} + 50x_{21} + 15x_{31} \leq 100y_1$$

$$100x_{12} + 50x_{22} + 15x_{32} \leq 75y_2$$

$$100x_{13} + 50x_{23} + 15x_{33} \leq 200y_3$$

Solutie: Exista moduri diferite de a codifica variabilele problemei:

(Variant 1**)** ca numere intregi

(Variant 2**)** ca numere reale

(Variant 3**)** ca valori booleene

(Variant 4**)** folosind constrangeri `assert-soft`.

Varianta 1.Declaram fiecare variabila ca numar intreg, e.g.:

```
(declare-const x11 Int)
```

Trebuie sa impunem de asemenea ca variabilele sunt 0/1, de exemplu:

```
(assert (and (>= x11 0) (>= x12 0) (>= x13 0) (>= x21 0) ...
(assert (and (<= y1 1) (<= y2 1) (<= y3 1)))
```

O alta varianta de codificare a intregilor 0/1 este:

```
(assert (or (>= x11 0) (<= x11 1)
(assert (or (>= x12 0) (<= x12 1)
...
```

Constragerile de tipul 2 pot fi codificate in 2 moduri. De exemplu:

```
(assert (and (>= y1 x11) (>= y1 x21) (>= y1 x31)))
...
```

Sau ca:

```
(assert (implies (or (= x11 1) (= x21 1) (= x31 1) ) (= y1 1) )
...
```

Varianta 2. Declaram fiecare variabila ca numar real. Constrangerile sunt la fel ca si codificarea cu numere intregi.
Varianta 3. Declaram variabila ca variabila booleana. Constrangerile de capabilitate/hardware permit doar variabile intregi/reale, deci trebuie sa transformam variabilele booleene in intregi/reale. Aceasta se poate realiza declarand o variabila astfel:

```
(define-fun bool_to_int ((b Bool)) Int (ite b 1 0) )
```

si folosind operatorul de cast pentru a transforma in variabilele in int/real, e.g.
`(assert (<= (+ (* 100 (bool_to_int x11)) (* 50 (bool_to_int x21)) (* 15 (bool_to_int x31)))`
Varianta 4. O alta varianta este de a folosi constrangeri `assert-soft`. Intr-o problema de optimizare, pot exista constrangeri care trebuie sa fie indeplinite (constrangeri de tip "hard") in timp ce celelalte pot fi violate (constrangeri de tip "soft").

Z3 ofera cadru pentru constrangeri "soft". Comanda `(assert-soft formula :weight numeral)` declara o constragere "soft" cu pondere. Ponderea trebuie sa fie un numar natural pozitiv si este optionala. Daca este omisa, atunci este setata la 1. De exemplu: `(declare-const a1 Bool)`

```
(declare-const a2 Bool)
(declare-const a3 Bool)
(assert-soft a1 :weight 0.1)
(assert-soft a2 :weight 1.0)
(assert-soft a3 :weight 1)
(assert-soft (or (not a1) (not a2) (not a3)))
```

```
(check-sat)
(get-model)
```

De asemenea, este posibil sa se declare mai multe clase de constrangeri "soft". Pentru a face acest lucru, se utilizeaza o eticheta optionala pentru a diferentia clasele de constrangeri "soft". De exemplu: `(declare-const a Bool)`

```
(declare-const b Bool)
(declare-const c Bool)
(assert-soft a :weight 1 :id A)
(assert-soft b :weight 2 :id B)
(assert-soft c :weight 3 :id A)
(assert (= a c))
(assert (not (and a b)))
(check-sat)
(get-model)
(get-objectives)
```

Pentru problema noastra, constrangerile soft pot fi utilizate pentru a codifica obiectivele de optimizare:

```
(assert-soft (not y1) :id num_servers)
(assert-soft (not y2) :id num_servers)
(assert-soft (not y3) :id num_servers)

(assert-soft (not y1) :id costs :weight 10)
(assert-soft (not y2) :id costs :weight 5)
(assert-soft (not y3) :id costs :weight 20)
```

Comanda `assert-soft` reprezinta MaxSMT (maximizeaza numarul de constrangeri care pot fi satisfacute) care incearca sa maximizeze suma ponderata a expresiilor booleene apartinand aceluiasi id. Deoarece facem minimizare, este necesara negarea pentru a profita de suportul MaxSMT.

Cerinte:

1p Scrieti problema de optimizare liniara ce trebuie sa o rezolvati (variabilele de decizie, constrangerile, functia obiectiv).

1p Scrieti fisierul SMT-LIB pentru fiecare dintre cele 4 codificari si rulati-l.

1p Pentru fiecare dintre cele 4 variante, rezolvati problema intr-un API al solverului Z3, de preferinta Python API. Afisati rezultatul. Programul trebuie sa genereaza si fisierul in formatul SMT-LIB corespunzator.

1p Exista vreo diferenta in ordinea in care sunt considerate functiile obiectiv? Explicati de ce.

1p Utilizati diferite variante de optimizare multicriteriala. Explicati alegerile si rezultatul.

2 Problema de Planificare a Magazinului (Job Shop Scheduling Problem – JSSP)

JSSP este o problema de optimizare in informatica si cercetare operationala, in care joburile sunt alocate resurselor in anumite momente. Versiunea clasica a problemei este dupa cum urmeaza:

se dau n joburi J_1, J_2, ..., J_n cu timpi de procesare variati, care trebuie sa fie programate pe m masini cu putere de procesare variabila, in timp ce se doreste **minimizarea *makepanului***. *Makespan* este lungimea totala a planificarii (adica, cand toate joburile au terminat de procesat). In plus, sunt implicate urmatoarele constrangeri:

- *Precedenta.* Taskurile unui job trebuie sa fie executate secvential.
- *Resurse.* Masinile executa cel mult un job la un moment dat.

Figure 1: Exemplu de JSSP

d_{ij}	Machine 1	Machine 2
Job 1	2	1
Job 2	3	1
Job 3	2	3

max = 8

Fie problema din Figura 1. Fie d_{ij} durata jobului i pe masina j. De exemplu $d_{11} = 2$ unitati de timp (time units TU), $d_{12} = 1$ TU, etc. Makespanul maximal este 8 TU.

2p Descrieri matematic problemele ce se rezolva la punctele urmatoare.

1p Gasiti o planificare potrivita pt. problema din Fig. 1 folosind Z3 SMT solver. Este aceasta o planificare optimala (adica care sa minimize makespanul)?

1p Scrieti si rezolvati problema de optimizare care duce la minimizarea makespanului.

1p Desenati ambele planificari obtinute.