Lab2 - Ex1

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

Variabile de decizie:

- $x_{11}, x_{12}, x_{13}, x_{21}, x_{22}, x_{23}, x_{31}, x_{32}, x_{33}$ — unde x_{ij} denota faptul ca pe masina virtuala i este alocat server-ul j
- y_1, y_2, y_3 — unde y_i denota faptul ca server-ul i este dat in folosinta

Constrangeri:

1. Variabilele de decizie pot avea valori de 0 sau 1
   - $x_{ij} \in \{0, 1\}, \forall i, j \in \{1, 2, 3\}$
   - $y_i \in \{0, 1\}, \forall i \in \{1, 2, 3\}$
2. O masina virtuala se aloca doar unui server
   - $x_{i1} + x_{i2} + x_{i3} = 1, \forall i \in \{1, \ldots, 3\}$
3. Un server este activ daca gazduieste o masina virtuala
   - $(x_{1j} = 1) \bigvee (x_{2j} = 1) \bigvee (x_{3j} = 1) \implies (y_j = 1), \forall j \in \{1, ..., 3\})$
4. Constrangeri de capabilitate/hardware
   - $100 * x_{11} + 50 * x_{21} + 15 * x_{31} \leq 100 * y_1$
   - $100 * x_{12} + 50 * x_{22} + 15 * x_{32} \leq 75 * y_2$
   - $100 * x_{13} + 50 * x_{23} + 15 * x_{33} \leq 200 * y_3$

Functia obiectiv:

Minimizarea costul operational: $10 * y_1 + 5 * y_2 + 20 * y_3$

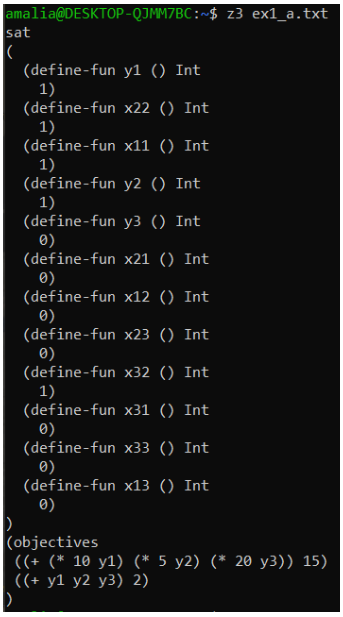
Minimizarea numarului de servere folosite: $y_1 + y_2 + y_3$

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

a) Codificare: numere intregi

```
; Variabile
(declare-fun x11 () Int)
(declare-fun x12 () Int)
(declare-fun x13 () Int)
(declare-fun x21 () Int)
(declare-fun x22 () Int)
(declare-fun x23 () Int)
(declare-fun x31 () Int)
(declare-fun x32 () Int)
(declare-fun x33 () Int)

(declare-fun y1 () Int)
(declare-fun y2 () Int)
(declare-fun y3 () Int)

; Constrangeri
;  1. Valori de 0|1
(assert (and (>= x11 0) (<= x11 1)))
(assert (and (>= x12 0) (<= x12 1)))
```

```
(assert (and (>= x13 0) (<= x13 1)))
(assert (and (>= x21 0) (<= x21 1)))
(assert (and (>= x22 0) (<= x22 1)))
(assert (and (>= x23 0) (<= x23 1)))
(assert (and (>= x31 0) (<= x31 1)))
(assert (and (>= x32 0) (<= x32 1)))
(assert (and (>= x33 0) (<= x33 1)))

(assert (or (>= y1 0) (<= y1 1)))
(assert (or (>= y2 0) (<= y2 1)))
(assert (or (>= y3 0) (<= y3 1)))

;  2. O masina virtuala se aloca doar unui server
(assert (= (+ x11 x12 x13) 1))
(assert (= (+ x21 x22 x23) 1))
(assert (= (+ x31 x32 x33) 1))

;  3. Un server este activ daca gazduieste o masina virtuala
(assert (and (>= y1 x11) (>= y1 x21) (>= y1 x31)))
(assert (and (>= y2 x12) (>= y2 x22) (>= y2 x32)))
(assert (and (>= y3 x13) (>= y3 x23) (>= y3 x33)))

;  4. Constrangeri de capabilitate/hardware
(assert (<= (+ (* 100 x11) (* 50 x21) (* 15 x31)) (* 100 y1)))
(assert (<= (+ (* 100 x12) (* 50 x22) (* 15 x32)) (* 75 y2)))
(assert (<= (+ (* 100 x13) (* 50 x23) (* 15 x33)) (* 200 y3)))

; Functia obiectiv
(minimize (+ (* 10 y1) (* 5 y2) (* 20 y3)))
(minimize (+ y1 y2 y3))

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

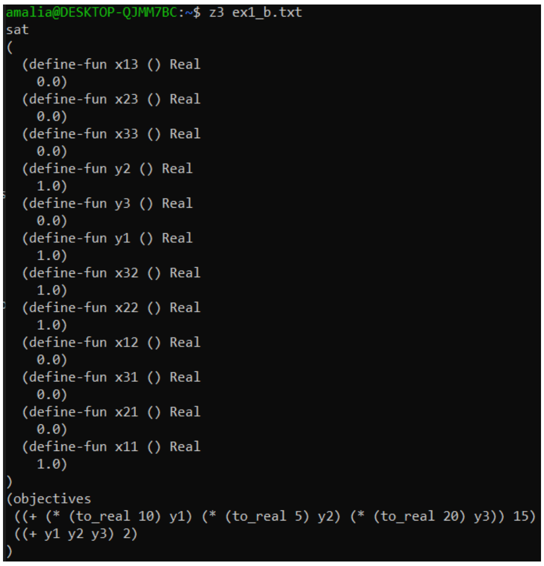


```
amalia@DESKTOP-QJMM7BC:~$ z3 ex1_a.txt
sat
(
  (define-fun y1 () Int
    1)
  (define-fun x22 () Int
    1)
  (define-fun x11 () Int
    1)
  (define-fun y2 () Int
    1)
  (define-fun y3 () Int
    0)
  (define-fun x21 () Int
    0)
  (define-fun x12 () Int
    0)
  (define-fun x23 () Int
    0)
  (define-fun x32 () Int
    1)
  (define-fun x31 () Int
    0)
  (define-fun x33 () Int
    0)
  (define-fun x13 () Int
    0)
)
(objectives
 ((+ (* 10 y1) (* 5 y2) (* 20 y3)) 15)
 ((+ y1 y2 y3) 2)
)
```

b) Codificare: numere reale

```
; Variabile
(declare-fun x11 () Real)
(declare-fun x12 () Real)
(declare-fun x13 () Real)
(declare-fun x21 () Real)
(declare-fun x22 () Real)
(declare-fun x23 () Real)
(declare-fun x31 () Real)
(declare-fun x32 () Real)
(declare-fun x33 () Real)

(declare-fun y1 () Real)
(declare-fun y2 () Real)
(declare-fun y3 () Real)

; Constrangeri
;  1. Valori de 0|1
(assert (or (= x11 0) (= x11 1)))
(assert (or (= x12 0) (= x12 1)))
(assert (or (= x13 0) (= x13 1)))
(assert (or (= x21 0) (= x21 1)))
(assert (or (= x22 0) (= x22 1)))
(assert (or (= x23 0) (= x23 1)))
(assert (or (= x31 0) (= x31 1)))
(assert (or (= x32 0) (= x32 1)))
(assert (or (= x33 0) (= x33 1)))

(assert (or (= y1 0) (= y1 1)))
(assert (or (= y2 0) (= y2 1)))
(assert (or (= y3 0) (= y3 1)))

;  2. O masina virtuala se aloca doar unui server
(assert (= (+ x11 x12 x13) 1))
(assert (= (+ x21 x22 x23) 1))
(assert (= (+ x31 x32 x33) 1))

;  3. Un server este activ daca gazduieste o masina virtuala
(assert (and (>= y1 x11) (>= y1 x21) (>= y1 x31)))
(assert (and (>= y2 x12) (>= y2 x22) (>= y2 x32)))
(assert (and (>= y3 x13) (>= y3 x23) (>= y3 x33)))

;  4. Constrangeri de capabilitate/hardware
(assert (<= (+ (* 100 x11) (* 50 x21) (* 15 x31)) (* 100 y1)))
(assert (<= (+ (* 100 x12) (* 50 x22) (* 15 x32)) (* 75 y2)))
(assert (<= (+ (* 100 x13) (* 50 x23) (* 15 x33)) (* 200 y3)))

; Functia obiectiv
(minimize (+ (* 10 y1) (* 5 y2) (* 20 y3)))
(minimize (+ y1 y2 y3))

(check-sat)
```

```
(get-model)
(get-objectives)
```

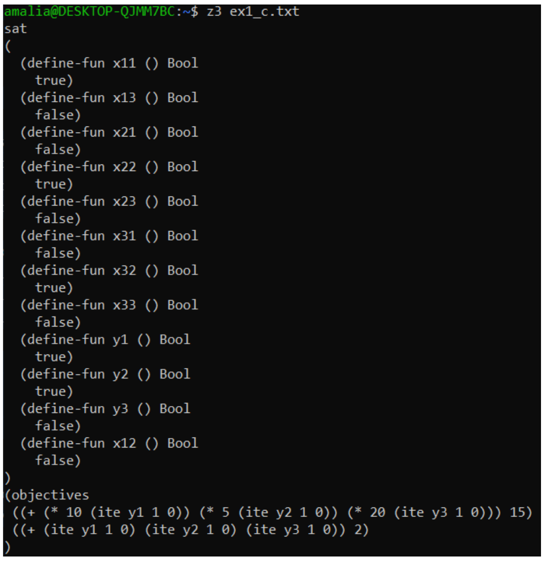


```
amalia@DESKTOP-QJMM7BC:~$ z3 ex1_b.txt
sat
(
  (define-fun x13 () Real
    0.0)
  (define-fun x23 () Real
    0.0)
  (define-fun x33 () Real
    0.0)
  (define-fun y2 () Real
    1.0)
  (define-fun y3 () Real
    0.0)
  (define-fun y1 () Real
    1.0)
  (define-fun x32 () Real
    1.0)
  (define-fun x22 () Real
    1.0)
  (define-fun x12 () Real
    0.0)
  (define-fun x31 () Real
    0.0)
  (define-fun x21 () Real
    0.0)
  (define-fun x11 () Real
    1.0)
)
(objectives
 ((+ (* (to_real 10) y1) (* (to_real 5) y2) (* (to_real 20) y3)) 15)
 ((+ y1 y2 y3) 2)
)
```

c) Codificare: valori booleene

```
; Variabile
(declare-fun x11 () Bool)
(declare-fun x12 () Bool)
(declare-fun x13 () Bool)
(declare-fun x21 () Bool)
(declare-fun x22 () Bool)
(declare-fun x23 () Bool)
(declare-fun x31 () Bool)
(declare-fun x32 () Bool)
(declare-fun x33 () Bool)

(declare-fun y1 () Bool)
(declare-fun y2 () Bool)
(declare-fun y3 () Bool)

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

; Constrangeri
;  1. Valori de 0|1
(assert (or x11 x12 x13 x21 x22 x23 x31 x32 x33 y1 y2 y3))

;  2. O masina virtuala se aloca doar unui server
(assert (= (+ (bool_to_int x11) (bool_to_int x12) (bool_to_int x13)) 1))
(assert (= (+ (bool_to_int x21) (bool_to_int x22) (bool_to_int x23)) 1))
(assert (= (+ (bool_to_int x31) (bool_to_int x32) (bool_to_int x33)) 1))

;  3. Un server este activ daca gazduieste o masina virtuala
(assert (and (>= (bool_to_int y1) (bool_to_int x11)) (>= (bool_to_int y1) (bool_to_int x
(assert (and (>= (bool_to_int y2) (bool_to_int x12)) (>= (bool_to_int y2) (bool_to_int x
(assert (and (>= (bool_to_int y3) (bool_to_int x13)) (>= (bool_to_int y3) (bool_to_int x

;  4. Constrangeri de capabilitate/hardware
```

```
(assert (<= (+ (* 100 (bool_to_int x11)) (* 50 (bool_to_int x21)) (* 15 (bool_to_int x31
(assert (<= (+ (* 100 (bool_to_int x12)) (* 50 (bool_to_int x22)) (* 15 (bool_to_int x32
(assert (<= (+ (* 100 (bool_to_int x13)) (* 50 (bool_to_int x23)) (* 15 (bool_to_int x33

; Functia obiectiv
(minimize (+ (* 10 (bool_to_int y1)) (* 5 (bool_to_int y2)) (* 20 (bool_to_int y3))))
(minimize (+ (bool_to_int y1) (bool_to_int y2) (bool_to_int y3)))

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

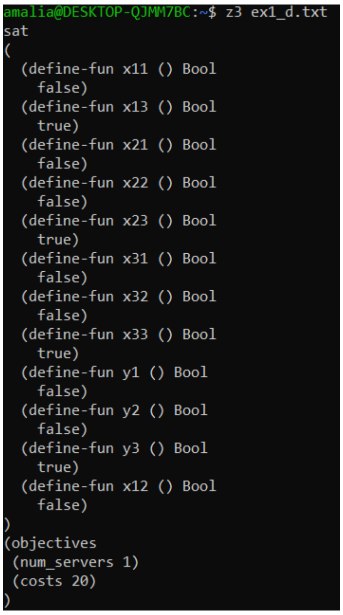


```
amalia@DESKTOP-QJMM7BC:~$ z3 ex1_c.txt
sat
(
  (define-fun x11 () Bool
    true)
  (define-fun x13 () Bool
    false)
  (define-fun x21 () Bool
    false)
  (define-fun x22 () Bool
    true)
  (define-fun x23 () Bool
    false)
  (define-fun x31 () Bool
    false)
  (define-fun x32 () Bool
    true)
  (define-fun x33 () Bool
    false)
  (define-fun y1 () Bool
    true)
  (define-fun y2 () Bool
    true)
  (define-fun y3 () Bool
    false)
  (define-fun x12 () Bool
    false)
)
(objectives
 ((+ (* 10 (ite y1 1 0)) (* 5 (ite y2 1 0)) (* 20 (ite y3 1 0))) 15)
 ((+ (ite y1 1 0) (ite y2 1 0) (ite y3 1 0)) 2)
)
```

d) Codificare: constrangeri assert-soft

```
; Variabile
(declare-fun x11 () Bool)
(declare-fun x12 () Bool)
(declare-fun x13 () Bool)
(declare-fun x21 () Bool)
(declare-fun x22 () Bool)
(declare-fun x23 () Bool)
(declare-fun x31 () Bool)
(declare-fun x32 () Bool)
(declare-fun x33 () Bool)

(declare-fun y1 () Bool)
(declare-fun y2 () Bool)
(declare-fun y3 () Bool)

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

; Constrangeri
;  1. Valori de 0|1
(assert (or x11 x12 x13 x21 x22 x23 x31 x32 x33 y1 y2 y3))

;  2. O masina virtuala se aloca doar unui server
(assert (= (+ (bool_to_int x11) (bool_to_int x12) (bool_to_int x13)) 1))
```

```
(assert (= (+ (bool_to_int x21) (bool_to_int x22) (bool_to_int x23)) 1))
(assert (= (+ (bool_to_int x31) (bool_to_int x32) (bool_to_int x33)) 1))

;  3. Un server este activ daca gazduieste o masina virtuala
(assert (and (>= (bool_to_int y1) (bool_to_int x11)) (>= (bool_to_int y1) (bool_to_int x
(assert (and (>= (bool_to_int y2) (bool_to_int x12)) (>= (bool_to_int y2) (bool_to_int x
(assert (and (>= (bool_to_int y3) (bool_to_int x13)) (>= (bool_to_int y3) (bool_to_int x

;  4. Constrangeri de capabilitate/hardware
(assert (<= (+ (* 100 (bool_to_int x11)) (* 50 (bool_to_int x21)) (* 15 (bool_to_int x31
(assert (<= (+ (* 100 (bool_to_int x12)) (* 50 (bool_to_int x22)) (* 15 (bool_to_int x32
(assert (<= (+ (* 100 (bool_to_int x13)) (* 50 (bool_to_int x23)) (* 15 (bool_to_int x33

(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)

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



```
amalia@DESKTOP-QJMM7BC:~$ z3 ex1_d.txt
sat
(
  (define-fun x11 () Bool
    false)
  (define-fun x13 () Bool
    true)
  (define-fun x21 () Bool
    false)
  (define-fun x22 () Bool
    false)
  (define-fun x23 () Bool
    true)
  (define-fun x31 () Bool
    false)
  (define-fun x32 () Bool
    false)
  (define-fun x33 () Bool
    true)
  (define-fun y1 () Bool
    false)
  (define-fun y2 () Bool
    false)
  (define-fun y3 () Bool
    true)
  (define-fun x12 () Bool
    false)
)
(objectives
 (num_servers 1)
 (costs 20)
)
```

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

Da, exista o diferenta in ordinea in care sunt considerate functiile obiectiv. Daca schimbam ordinea lor, va fi prioritizata constrangerea de a minimiza numarul de servere, deci rezultatul nostru poate fi mai costisitor decat daca am prioritiza minimizarea costului. Depinde ce fel de rezultat preferam, cost mai mic sau mai putine servere.

5. Utilizati diferite variante de optimizare multicriteriala. Explicati alegerile si rezultatul.

Optimizarea criteriala se refera la procesul de gasire a solutiolor care optimizeaza simultan mai multe criterii care pot fi contradictorii sau competitoare intre ele sau contradictorii.

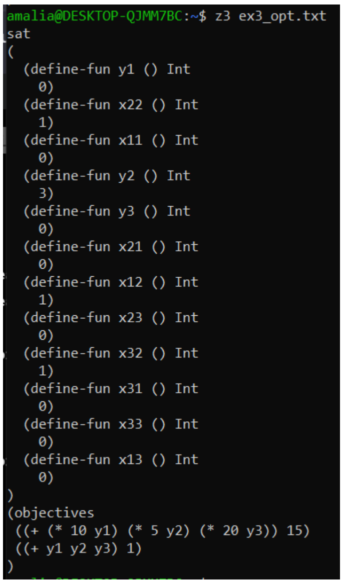
Putem folosi optiunea `(set-option :opt.priority NAME)` pt a seta prioritatea pentru ptimizarea variabilor

- (set-option :opt.priority lex) → ordinea in care au fost adaugate
- (set-option :opt.priority box) → in cadrul unui domeniu variabil, exista uneori sub-domenii mai mici in care se pot gasi solutii optime. Ideea este de a explora mai intai aceste sub-dimenii inainte de a se concentra pe variabilele individuale.

```
(set-option :opt.priority lex)
;(set-option :opt.priority lex)

; Variabile
(declare-fun x11 () Int)
(declare-fun x12 () Int)
(declare-fun x13 () Int)
(declare-fun x21 () Int)
(declare-fun x22 () Int)
(declare-fun x23 () Int)
(declare-fun x31 () Int)
(declare-fun x32 () Int)
(declare-fun x33 () Int)

(declare-fun y1 () Int)
(declare-fun y2 () Int)
(declare-fun y3 () Int)

; Constrangeri
;  1. Valori de 0|1
(assert (and (>= x11 0) (<= x11 1)))
(assert (and (>= x12 0) (<= x12 1)))
(assert (and (>= x13 0) (<= x13 1)))
(assert (and (>= x21 0) (<= x21 1)))
(assert (and (>= x22 0) (<= x22 1)))
(assert (and (>= x23 0) (<= x23 1)))
(assert (and (>= x31 0) (<= x31 1)))
(assert (and (>= x32 0) (<= x32 1)))
(assert (and (>= x33 0) (<= x33 1)))

(assert (or (>= y1 0) (<= y1 1)))
(assert (or (>= y2 0) (<= y2 1)))
(assert (or (>= y3 0) (<= y3 1)))

;  2. O masina virtuala se aloca doar unui server
(assert (= (+ x11 x12 x13) 1))
(assert (= (+ x21 x22 x23) 1))
(assert (= (+ x31 x32 x33) 1))

;  3. Un server este activ daca gazduieste o masina virtuala
(assert (and (>= y1 x11) (>= y1 x21) (>= y1 x31)))
(assert (and (>= y2 x12) (>= y2 x22) (>= y2 x32)))
(assert (and (>= y3 x13) (>= y3 x23) (>= y3 x33)))

;  4. Constrangeri de capabilitate/hardware
(assert (<= (+ (* 100 x11) (* 50 x21) (* 15 x31)) (* 100 y1)))
```

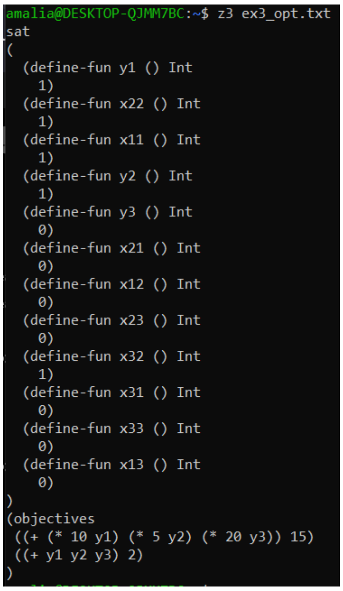
```
(assert (<= (+ (* 100 x12) (* 50 x22) (* 15 x32)) (* 75 y2)))
(assert (<= (+ (* 100 x13) (* 50 x23) (* 15 x33)) (* 200 y3)))

; Functia obiectiv
(minimize (+ (* 10 y1) (* 5 y2) (* 20 y3)))
(minimize (+ y1 y2 y3))

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

Cu optiunea ‘box’



```
amalia@DESKTOP-QJMM7BC:~$ z3 ex3_opt.txt
sat
(
  (define-fun y1 () Int
    0)
  (define-fun x22 () Int
    1)
  (define-fun x11 () Int
    0)
  (define-fun y2 () Int
    3)
  (define-fun y3 () Int
    0)
  (define-fun x21 () Int
    0)
  (define-fun x12 () Int
    1)
  (define-fun x23 () Int
    0)
  (define-fun x32 () Int
    1)
  (define-fun x31 () Int
    0)
  (define-fun x33 () Int
    0)
  (define-fun x13 () Int
    0)
)
(objectives
 ((+ (* 10 y1) (* 5 y2) (* 20 y3)) 15)
 ((+ y1 y2 y3) 1)
)
```

Cu optiunea ‘lex’



```
amalia@DESKTOP-QJMM7BC:~$ z3 ex3_opt.txt
sat
(
  (define-fun y1 () Int
    1)
  (define-fun x22 () Int
    1)
  (define-fun x11 () Int
    1)
  (define-fun y2 () Int
    1)
  (define-fun y3 () Int
    0)
  (define-fun x21 () Int
    0)
  (define-fun x12 () Int
    0)
  (define-fun x23 () Int
    0)
  (define-fun x32 () Int
    1)
  (define-fun x31 () Int
    0)
  (define-fun x33 () Int
    0)
  (define-fun x13 () Int
    0)
)
(objectives
 ((+ (* 10 y1) (* 5 y2) (* 20 y3)) 15)
 ((+ y1 y2 y3) 2)
)
```