- **Homework H_1**: Find the minimum for the following functions: De Jong 1, Schwefel's, Rastrigin's, Michalewicz's using the Hill Climbing (the first improvement, best improvement and worst improvement variants) and Simulated Annealing algorithms (hybridising one of the Hill Climbing variants).
  Analyse the 5-, 10- and 30-dimensional versions of all functions. (Perhaps run the 100-dimensional version as well, as an upper bound). Use a precision of at least 5 decimal places after 0.
  Soft deadline: week 5 (-10% points for each week of delay)
  Send the homework: source code without binary object and .txt or .pdf file, in a .zip archive (no other archive format allowed) to my e-mail, with a subject written like: [GA]_Name_Surname_Group_T1

2.1 De Jong's function 1

The simplest test function is De Jong's function 1. It is also known as sphere model. It is continuos, convex and unimodal.

function definition:

$$f_1(x) = \sum_{i=1}^{n} x_i^2 \qquad -5.12 \le x_i \le 5.12$$

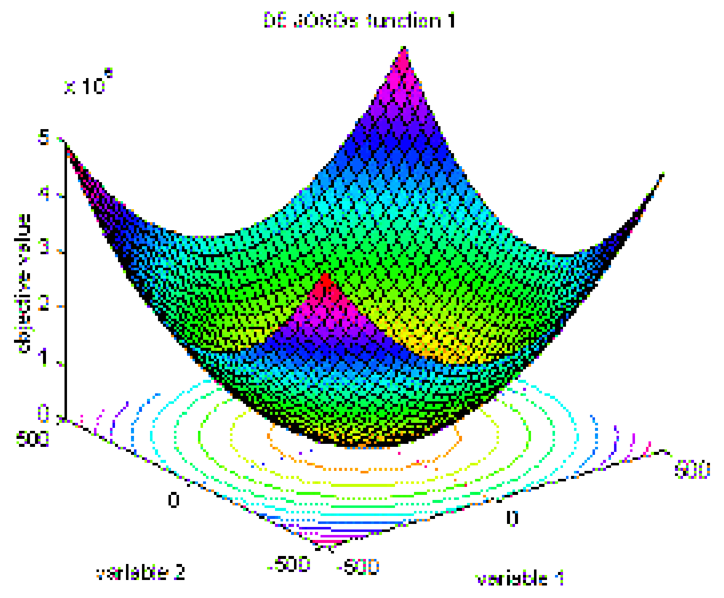
```
f1(x)=sum(x(i)^2), i=1:n, -5.12<=x(i)<=5.12.
```

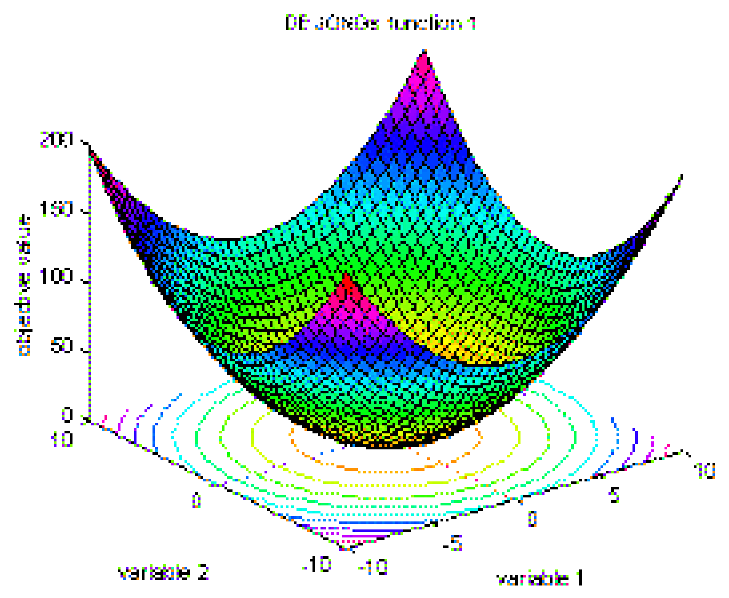
global minimum:

```
f(x)=0, x(i)=0, i=1:n.
```

This function is implemented in objfun1 .

Fig. 2-1: Visualization of De Jong's function 1 using different domains of the variables; however, both graphics look similar, just the scaling changed; left: surf plot of the function in a very large area from -500 to 500 for each of both variables, right: the function at a smaller area from -10 to 10





Documentation of objfun1

Function Synopsis

```
ObjVal = objfun1(Chrom, P1, P2);
```

Help text

```
OBJective function for de jong's FUNction 1

This function implements the DE JONG function 1.

Syntax:  ObjVal = objfun1(Chrom, P1, P2)

Input parameters:
   Chrom     - Matrix containing the chromosomes of the current
               population. Each row corresponds to one individual's
               string representation.
               if Chrom == [NaN xxx], then special values will be returned
               xxx == 1 (or []) return boundaries
               xxx == 2 return title
               xxx == 3 return value of global minimum
   P1, P2    - Additional parameter

Output parameters:
   ObjVal    - Column vector containing the objective values of the
               individuals in the current population.
               if called with Chrom == [NaN xxx], then ObjVal contains
               xxx == 1, matrix with the boundaries of the function
               xxx == 2, text for the title of the graphic output
               xxx == 3, value of global minimum

See also: objfun1a, objfun1b, objfun2, objfun6, objfun7, objfun8, objfun9, objfun10, initfun1
```

2.7 Schwefel's function 7

Schwefel's function [Sch81] is deceptive in that the global minimum is geometrically distant, over the parameter space, from the next best local minima. Therefore, the search algorithms are potentially prone to convergence in the wrong direction.

function definition:

$$f_7(x) = \sum_{i=1}^{n} -x_i \cdot \sin\left(\sqrt{|x_i|}\right) \qquad -500 \le x_i \le 500$$

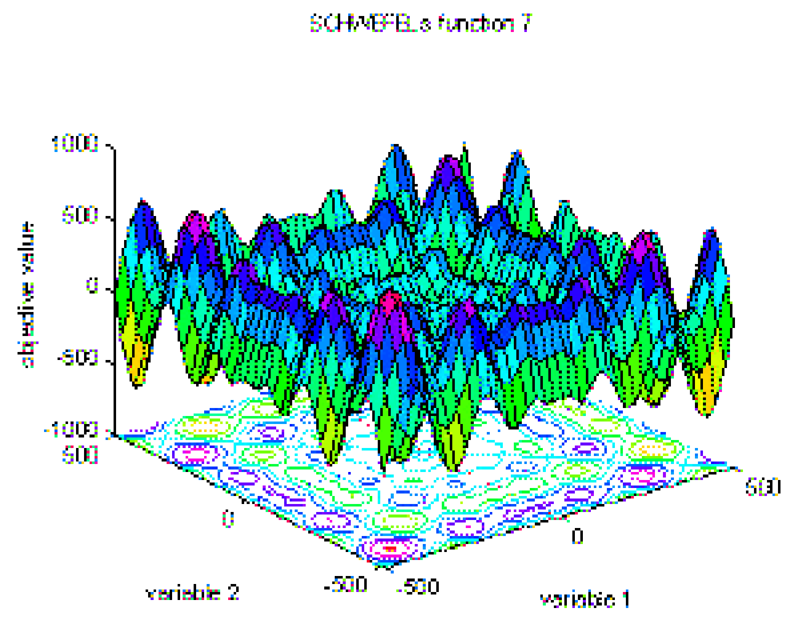
```
f7(x)=sum(-x(i)·sin(sqrt(abs(x(i))))), i=1:n; -500<=x(i)<=500.
```

global minimum:

```
f(x)=-n·418.9829; x(i)=420.9687, i=1:n.
```

This function is implemented in objfun7.

Fig. 2-7: Visualization of Schwefel's function; surf plot in an area from -500 to 500



Documentation of objfun7

Function Synopsis

```
ObjVal = objfun7(Chrom, option);
```

Help text

```
 OBJective function for schwefel's FUNction

 This function implements the SCHWEFEL function 7.

 Syntax:  ObjVal = objfun7(Chrom, option)

 Input parameters:
    Chrom     - Matrix containing the chromosomes of the current
                population. Each row corresponds to one individual's
                string representation.
                if Chrom == [], then speziell values will be returned
    option    - if Chrom == [] and
                option == 1 (or []) return boundaries
                option == 2 return title
                option == 3 return value of global minimum

 Output parameters:
    ObjVal    - Column vector containing the objective values of the
                individuals in the current population.
                if called with Chrom == [], then ObjVal contains
                option == 1, matrix with the boundaries of the function
                option == 2, text for the title of the graphic output
                option == 3, value of global minimum
```

2.6 Rastrigin's function 6

Rastrigin's function is based on function 1 with the addition of cosine modulation to produce many local minima. Thus, the test function is highly multimodal. However, the location of the minima are regularly distributed.

function definition:

$$f_6(x) = 10\cdot n + \sum_{i=1}^{n}\left(x_i^2 - 10\cdot\cos(2\cdot\pi\cdot x_i)\right) \qquad -5.12 \le x_i \le 5.12$$

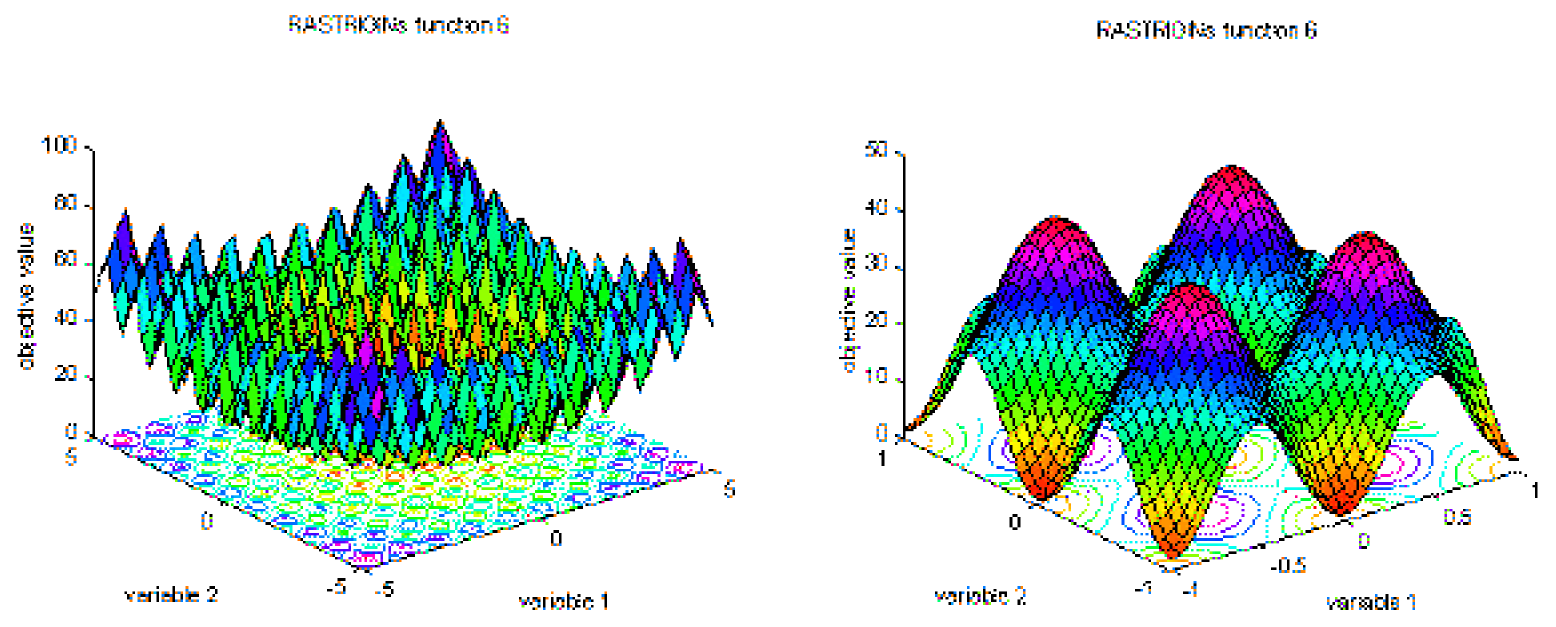
```
f6(x)=10·n+sum(x(i)^2-10·cos(2·pi·x(i))), i=1:n; -5.12<=x(i)<=5.12.
```

global minimum:

```
f(x)=0; x(i)=0, i=1:n.
```

This function is implemented in objfun6.

Fig. 2-6: Visualization of Rastrigin's function; left: surf plot in an area from -5 to 5, right: focus around the area of the global optimum at [0, 0] in an area from -1 to 1



Documentation of objfun6

Function Synopsis

```
ObjVal = objfun6(Chrom, option);
```

Help text

```
OBJective function for rastrigins FUNction 6

This function implements the RASTRIGIN function 6.

Syntax:  ObjVal = objfun6(Chrom, option)

Input parameters:
   Chrom     - Matrix containing the chromosomes of the current
               population. Each row corresponds to one individual's
               string representation.
               if Chrom == [], then speziell values will be returned
   option    - if Chrom == [] and
               option == 1 (or []) return boundaries
               option == 2 return title
               option == 3 return value of global minimum

Output parameters:
   ObjVal    - Column vector containing the objective values of the
               individuals in the current population.
               if called with Chrom == [], then ObjVal contains
               option == 1, matrix with the boundaries of the function
               option == 2, text for the title of the graphic output
               option == 3, value of global minimum
```

2.12 Michalewicz's function 12

PREV NEXT TOP

The Michalewicz function [Mic92] is a multimodal test function (n! local optima). The parameter m defines the "steepness" of the valleys or edges. Larger m leads to more difficult search. For very large m the function behaves like a needle in the haystack (the function values for points in the space outside the narrow peaks give very little information on the location of the global optimum).

function definition:

$$f_{12}(x) = -\sum_{i=1}^{n} \sin(x_i)\cdot\left(\sin\left(\frac{i\cdot x_i^2}{\pi}\right)\right)^{2\cdot m} \qquad i=1:n,\ m=10,\ 0 \le x_i \le \pi$$

```
 f12(x)=-sum(sin(x(i))·(sin(i·x(i)^2/pi))^(2·m)), i=1:n, m=10
  0<=x(i)<=pi.
```

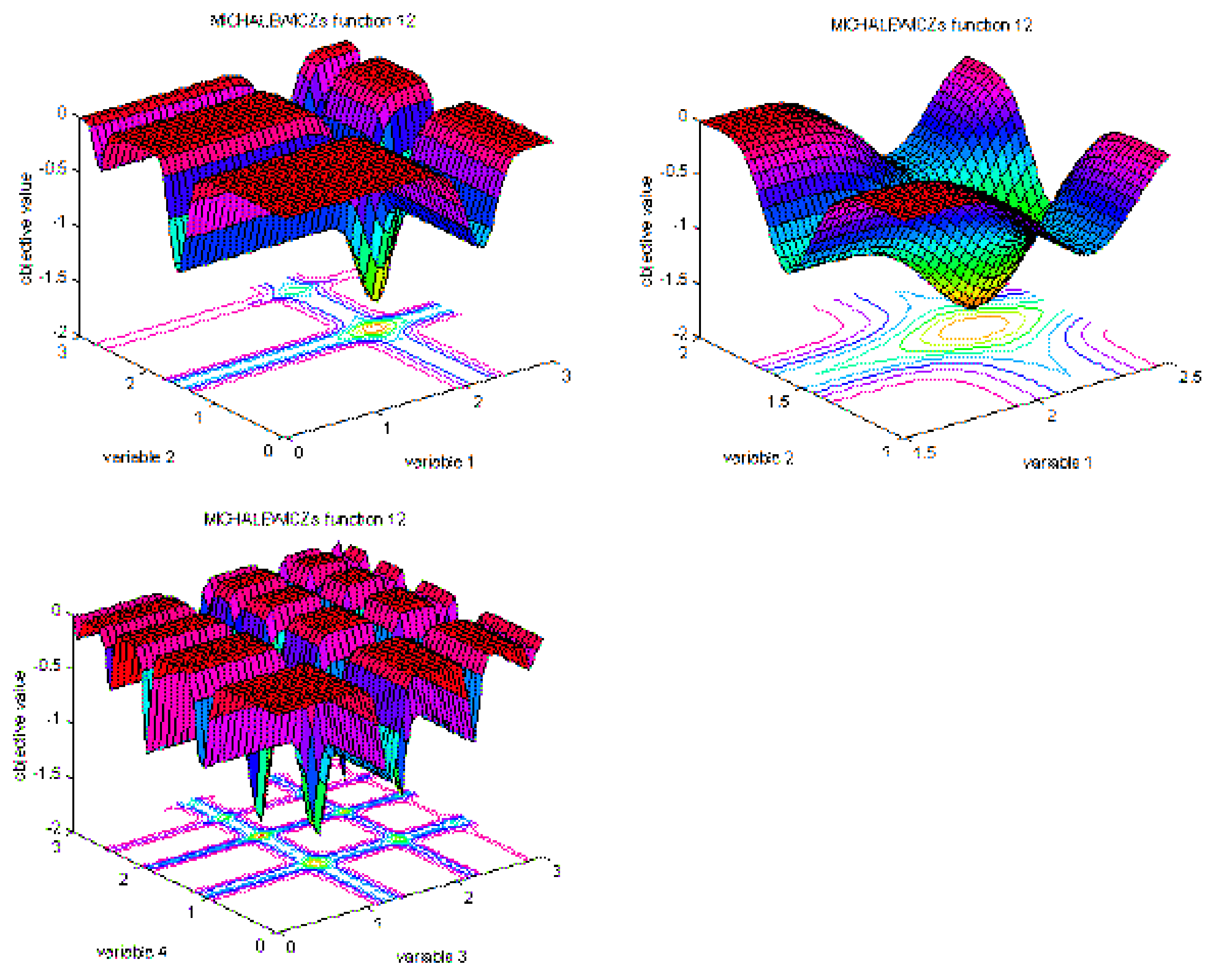
global minimum:

```
 f(x)=-4.687 (n=5); x(i)=???, i=1:n.
  f(x)=-9.66 (n=10); x(i)=???, i=1:n.
```

This function is implemented in objfun12.

The first two graphics below represent a global and a local view to Michalewicz's function, both for the first two variables. The third graphic on the right side displays the function using the third and fourth variable, the first two variables were set to 0. By comparing the left and the right graphic the increasing difficulty of the function can be seen. As higher the dimension as more valleys are introduced into the function.

Fig. 2-12: Visualization of Michalewicz's function; top left: surf plot in an area from 0 to 3 for the first and second variable, right: area around the optimum, bottom left: same as top left for the third and fourth variable, variable 1 and 2 are set 0



Documentation of objfun12

Function Synopsis

```
ObjVal = objfun12(Chrom, option);
```

Help text

```
OBJective function for michalewicz's function 12

This function implements the MICHALEWICZ function 12.

Syntax:  ObjVal = objfun12(Chrom, option)

Input parameters:
   Chrom     - Matrix containing the chromosomes of the current
               population. Each row corresponds to one individual's
               string representation.
               if Chrom == [], then special values will be returned
   option    - if Chrom == [] and
               option == 1 (or []) return boundaries
               option == 2 return title
               option == 3 return value of global minimum
               if Chrom is not empty, option can optionally contain
               the parameter m, which defines the stepness of the edges

Output parameters:
   ObjVal    - Column vector containing the objective values of the
               individuals in the current population.
               if called with Chrom == [], then ObjVal contains
               option == 1, matrix with the boundaries of the function
               option == 2, text for the title of the graphic output
               option == 3, value of global minimum

See also: objfun1a, objfun1b, objfun2, objfun6, objfun7, objfun8, objfun9, objfun10, initfun1

Examples:
   ObjVal = objfun12(Chrom);
   ObjVal = objfun12(Chrom, 10);

Reference:
```

(ro) Notiuni: Metode traiectorie

Hill Climbing

- metoda iterativa ce realizeaza o cautare locala
- denumita si *Imbunatatire iterativa (Iterative Improvement)*
- este utilizata varianta iterata (Iterated Hill Climbing) in care HC este restartat, pentru a mari gradul de explorare a spatiului de cautare

Pseudocod:

```
t := 0
initialize best
repeat
    local := FALSE
    select a candidate solution (bitstring) $v_c$ at random
    evaluate $v_c$
    repeat
        $v_n$ := Improve(Neghborhood($v_c$))
        if eval($v_n$) is better than eval($v_c$)
        then $v_c$ := $v_n$
        else local := TRUE
    until local
    t := t + 1
    if $v_c$ is better than best
    then best := $v_c$
until t = MAX
```

Functia *Improve* cauta in vecinatate prima solutie candidat care e mai buna decat solutia curenta sau cea mai buna solutie (first improvement sau best improvement).

Simulated Annealing

- Meta-euristica de tip traiectorie care permite o mai buna explorare a spatiului si iesirea din puncte de optim local, dand posibilitatea vizitarii unor solutii de calitate mai slaba decat cea curenta.

Pseudocod:

```
t := 0
initialize the temperature T
select a current candidate solution (bitstring) $v_c$ at random
evaluate $v_c$
repeat
    repeat
        select at random $v_n$: a neighbor of $v_c$
        if eval($v_n$) is better than eval($v_c$)
            then $v_c$ := $v_n$
            else if random[0,1) < exp(-|eval($v_n$) - eval($v_c$)| / T)
                then $v_c$ := $v_n$
    until (termination-condition)
    T := g(T; t)
    t := t + 1
until (halting-criterion)
```

Functia g(T; t) de modificare a "temperaturii" T va asigura o scadere treptata a acesteia cu fiecare iteratie (de exemplu, prin inmultirea cu o valoare subunitara).

Reprezentarea solutiilor:

- vectori de numere reale
- siruri binare: spatiul de cautare se va disctretiza pana la o anumita precizie 10^{-d}.
  Un interval [*a*, *b*] va fi impartit in `N = (b - a) * `10^d subintervale egale.
  Pentru a putea reprezenta cele `(b - a) * `10^d valori, este nevoie de un numar n = `parte_intreaga_superioara(`$\log_2$`(N))` de biti.
  Lungimea sirului de biti care reprezinta o solutie candidat va fi suma lungimilor reprezentarilor pentru fiecare parametru al functiei de optimizat.
  In momentul evaluarii solutiei (apelul functiei de optimizat) este necesara decodificarea fiecarui parametru reprezentat ca sir de biti in numar real, dupa formula:
  $X_{real} = a + decimal(x_{biti}) * (b - a) / (2^n - 1)$
  !!! Nu folositi reprezentari de forma vector de vectori de biti. Folositi un simplu vector de biti.
  Adica, reprezentati o solutie candidat ca un vector de biti, nu o matrice de biti.