Algoritmi genetici aplicați în planificarea și proiectarea rețelelor de telecomunicații

Horia Coman

Cuprins

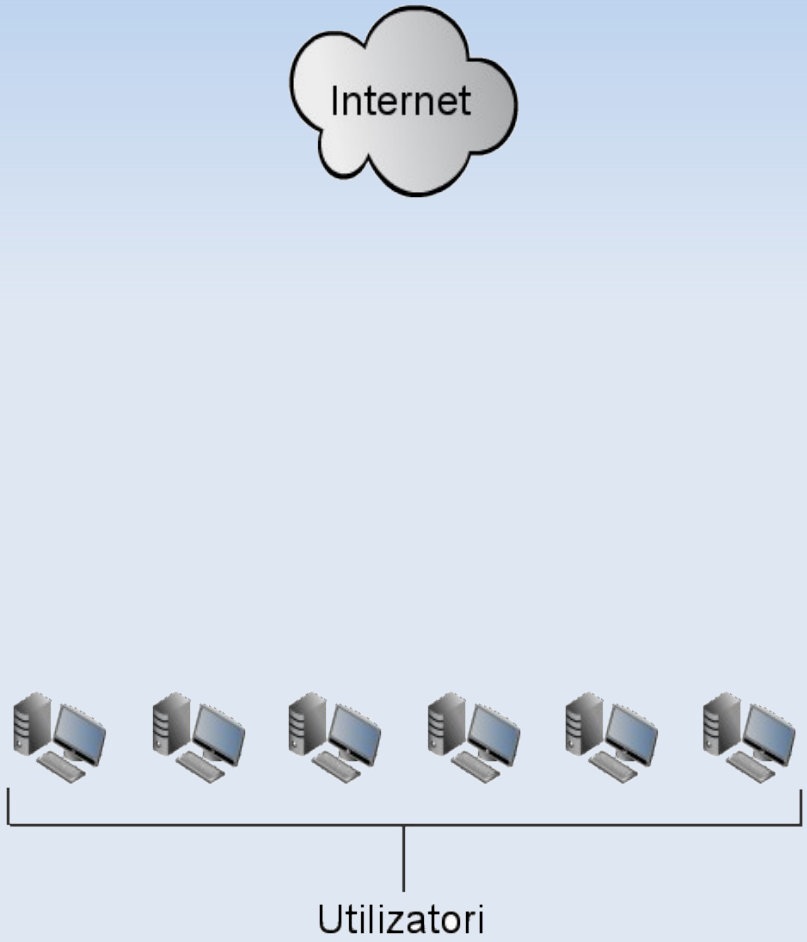
1. Problema. Formulare Matematica. Cautarea exhaustiva. Metode de Optimizare.
2. Detalii de implementare ale sistemului de optimizare.
3. Detalii algoritmi.
4. Concluzii.

Capitolul 1

1. **Problema. Formulare Matematica. Cautarea exhaustiva. Metode de Optimizare.**
2. Detalii de implementare ale sistemului de optimizare.
3. Detalii algoritmi.
4. Concluzii.

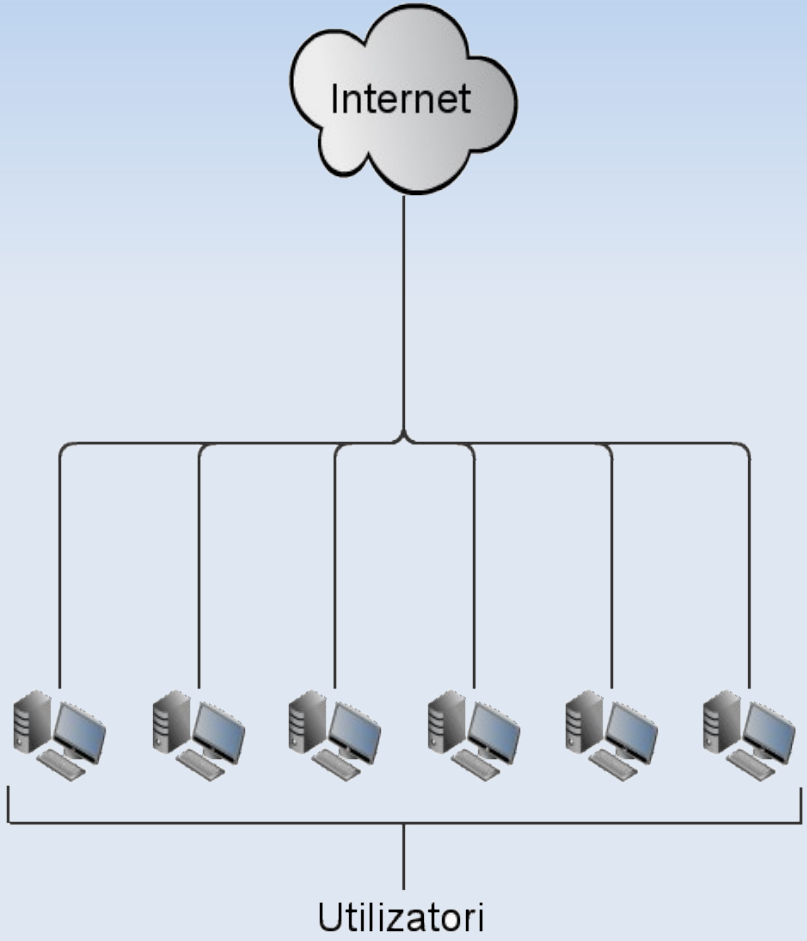
1.Problema

- **Q:** Ce inseamna o retea de acces?
- **A:** Retea de telecomunicatii ce permite accesul utilizatorilor la Internet.



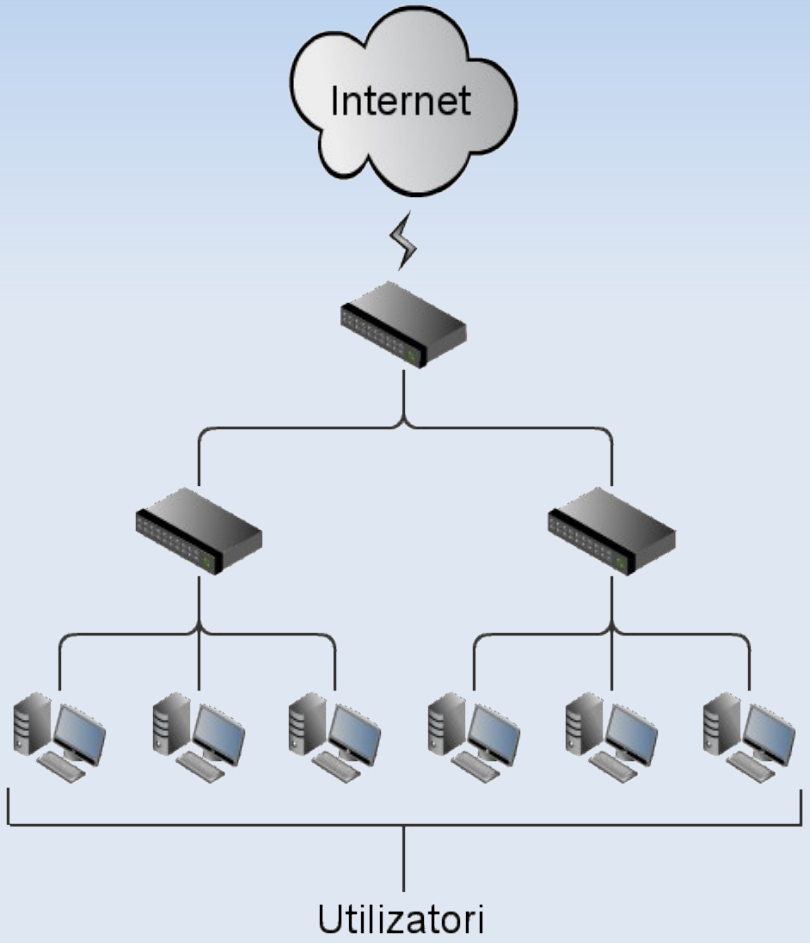
1.Problema

- **Q:** Cum arata o astfel de retea?
- **A:** Solutia cea mai simpla leaga toti utilizatorii direct la Internet.



1.Problema

- Solutie simpla este impractica fizic.
- Retele de acces reale au o forma arborescenta. Contin **Noduri** si **Legaturi**.



1.Problema

Problema: Construirea unei retele de acces din **Noduri** si **Legaturi**, astfel incat **Utilizatorii** sa poata accesa **Internetul**. Mai mult, aceasta trebuie sa fie optima: sa aiba **costul/pretul minim**.

1.Problema

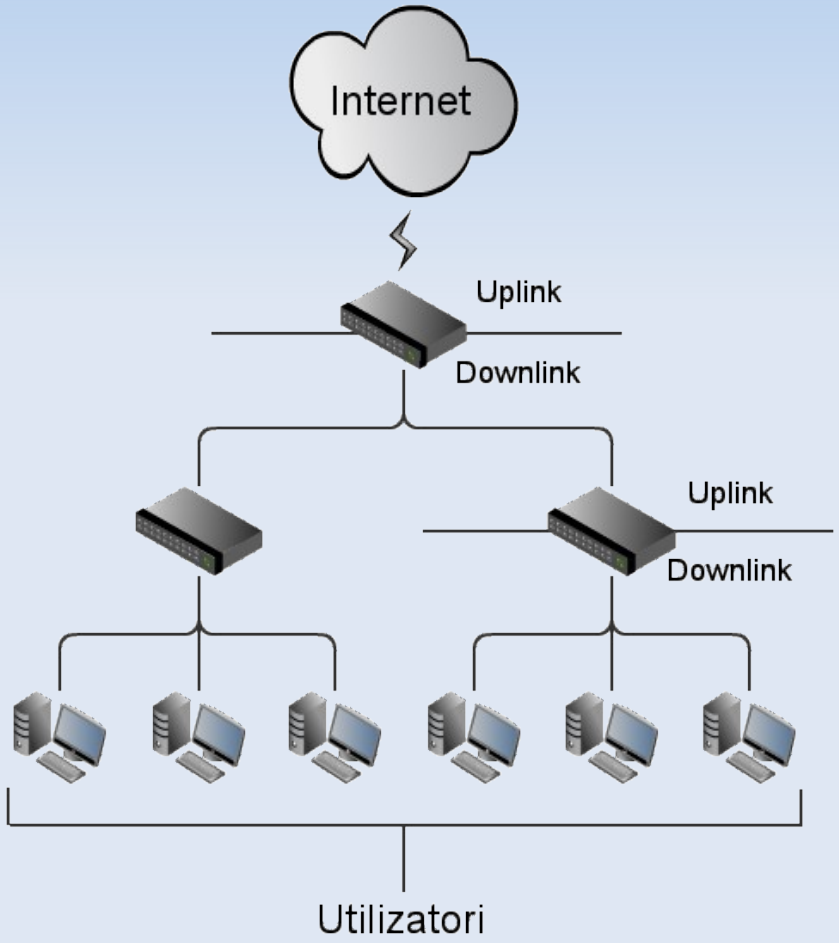
- **Nod**: entitate de agregare a comunicarii. Switch, Router etc.
- **Legatura**: entitate de transmisie a comunicarii. Ethernet, Fibra Optica, Wireless etc.
- Ne intereseaza mai mult Noduri decat Legaturi, in procesul de modelare.

1.Problema

- Despre Legaturi este deajuns sa stim ca exista si ca ”leaga” doua alte entitati.
- Nu ne intereseaza caracteristicile acestora.
- Decizie de modelare. Alte modele pot lua in considerare parameterii ale acestora.

1.Problema

- Modelarea unui Nod:
  - Cost
  - Numar de porturi Downlink, resp. Uplink
- **Port**: loc de legatura inspre/dispre nod.
- **Uplink**: spre Internet.
- **Downlink**: spre Utilizatori.



1.Problema

- Configuratie de Problema:
  - Structura matematica.
  - Componente:
    - Lista de utilizatori.
    - Lista de noduri.

```
CP : configurație de problema
CP = (CP:U,CP:N)
    unde CP:U : descrierea utilizatorilor
         CP:U = { ui : i de la 1 la ||CP:U|| }
         CP:N : descrierea nodurilor
         CP:N = { ni : i de la 1 la ||CP:N|| }
```

1.Problema

Problema Devine, considerand o configuratie de problema **CP**, alegerea unui numar de **Noduri** si **Legaturi** intre acestea, astfel incat reteaua rezultanta sa **lege toti utilizatorii** si sa fie de **cost minim**.

1.Problema

- Pentru o **CP** oarecare, exista o multitudine de retele de acces: **Spatiul Retelelor de Acces peste CP**.

$$SP\_RA(CP)$$

- O solutie este o lista de noduri (discutie in 3.).

$$ra(CP) = (n1,n2,n3,n4)$$
$$\text{unde } n1,n2,n3,n4 \in CP:N$$

- SP_RA(CP) este o **multime de liste de noduri.** Este **discret** si **finit**.

1.Problema

- In final, putem definii o **functie cost**, peste **SP_RA(CP)**, ce modeleaza matematic pretul unei retele de acces.

$$cost(CP): SP_{RA}(CP) \rightarrow \mathbb{R}$$

$$cost(CP)(ra) = \sum_{unde\ n_i \in ra} cost(n_i)$$

1.Problema

- **Problema**, in final, este: fiind data o configuratie **CP**, sa se aleaga elementul din **SP_RA(CP)** astfel incat functia **cost(CP)** sa fie minima.

$$\min_{ra \in SP_{RA}(CP)} cost(CP)$$

1.Problema

- Element in plus in modelare: cerintele utilizatorilor.
- Modelate doar ca cerinte de **Bandwidth**: volum de trafic pe secunda atat **Download** cat si **Upload**.
- Pe langa existenta, utilizatorul mai este caracterizat de doua numere: Viteza de Download si Viteza de Upload.

```
u = (u:dl,u:ul)
```

1.Problema

- **Retea Valida**: retea de acces ce leaga toti Utilizatorii la Internet, permitand fiecaruia sa comunice la viteza maxima, in acelasi timp.
- **Retea Valida**: reteaua care indeplineste obligatiile contractuale catre utilizatori.
- Sunt mai putine astfel de retele, decat retele normale. Subspatiu in SP_RA(CP).

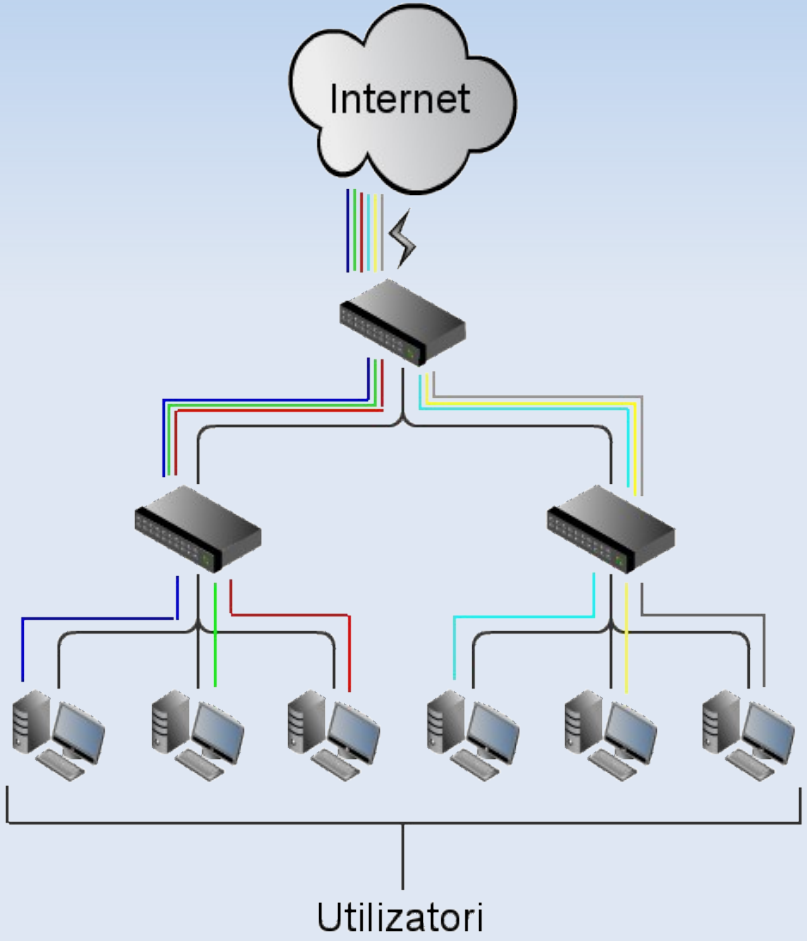
1.Problema

- Extindem definitia unui nod.
- Contine si **Volum maxim pe porturile de Downlink** si **Volum maxim pe porturile de uplink**.

```
n = (n:cost, # cost
     n:nrd,  # nr. porturi downlink
     n:d:dl, # download downlink
     n:d:ul, # upload downlink
     n:nru,  # nr. porturi uplink
     n:u:dl, # download uplink
     n:u:ul) # upload uplink
```

1.Problema

- Un nod agrega transmisii dinspre Downlink spre Uplink. Volumul de pe Uplink e suma Volumelor de pe fiecare port Downlink.
- Trebuie ca Volumul maxim admis sa nu fie depasit.



1.Problema

- Avem o noua structura pentru configuratia de Problema. Exemplu:

```
                    # ui:dl    ui:ul
cp = (cp:U = { u1  = (1.024,    0.128),
               u2  = (0.512,    0.128),
               u3  = (0.512,    0.064),
               u4  = (1.024,    0.256),
               u5  = (2.048,    1.024),
               u6  = (0.512,    0.128),
               u7  = (0.512,    0.256),
               u8  = (1.024,    0.256),
               u9  = (1.024,    0.128),
               u10 = (0.512,    0.512),
               u11 = (0.128,    0.064),
               u12 = (0.256,    0.128) },
                     # ni:cost  ni:nrd   ni:d:dl  ni:d:ul  ni:nru   ni:u:dl  ni:u:ul
      cp:N = { n1  = (100,      4,       10,      2,       1,       20,      2),
               n2  = (40,       2,       10,      4,       1,       20,      2),
               n3  = (150,      8,       10,      1,       2,       100,     10),
               n4  = (180,      4,       12,      4,       1,       40,      10),
               n5  = (250,      4,       20,      10,      1,       100,     30),
               n6  = (500,      4,       40,      40,      1,       150,     150) })
```

1.Problema

- In final, **Problema** este: Gasirea unei retele de **acces valida** si de **cost minim**, pornind de la o configuratie de problema **CP**.

```
CP : configurație de problema
CP = (CP:U,CP:N)
    unde CP:U : descrierea utilizatorilor
         CP:U = { ui : i de la 1 la ||CP:U|| }
         CP:N : descrierea nodurilor
         CP:N = { ni : i de la 1 la ||CP:N|| }


u = (u:dl,u:ul)


n = (n:cost, # cost
     n:nrd,  # nr. porturi downlink
     n:d:dl, # download downlink
     n:d:ul, # upload downlink
     n:nru,  # nr. porturi uplink
     n:u:dl, # download uplink
     n:u:ul) # upload uplink
```

$$cost(CP): SP_{RA}(CP) \rightarrow \mathbb{R}$$

$$cost(CP)(ra) = \sum_{unde\ n_i \in ra} cost(n_i)$$

$$\min_{ra \in SP_{RA}(CP)} cost(CP)$$

1.Problema

- Cautare Exhaustiva / Brute Force Search
  - Parcurge SP_RA(CP) si selecteaza reteaua de cost minim.
  - Gaseste intotdeauna reteaua optima.
  - Intuitiva.
  - Usor de implementat.
  - **Nefezabila pentru dimensiuni mari ale spatiului de cautare.**

1.Problema

- Ne salveaza asa-zisele metode de optimizare (metode stocastice de optimizare).
- **Cautare inteligenta** a spatiului bazate pe presupunerea ca **schimbari mici in structura solutiei produc schimbari mici de cost.**
- Am ales patru dintre multele metode:
  - Random Search
  - Hill Climbing
  - Evolution Strategy
  - Genetic Algorithm

1.Problema

- Vocabular Comun:
  - General
    - Individ
    - Populatie
    - Proces de Evolutie / Iteratie / Individ Optim
  - Referitor la individ:
    - Generare
    - Mutatie
    - Incrucisare
  - Referitor la populatie:
    - Selectie

1.Problema

- Primele doua sunt metode simple, de test/referinta. Metode neevolutive.
- Ultimele doua sunt metodele in care ne punem baza. Metode evolutive.
- Evolutiv: foloseste intreaga populatie pentru realizarea unui individ nou in procesul de evolutie.
- Conceptele de adineaori descriu metode evolutive. Adaptam metodele neevolutive astfel incat sa poata fi descrise totusi de acestea.

1.Problema

- Fiecare metoda poate fi descrisa folosind elementele generale (individ,populatie etc.) si un numar de operatori pe individ sau populatie.
- Random Search foloseste doar operatorul de generare.
- Genetic Algorithm ii foloseste pe toti.
- Detalii in **3.**

Capitolul 2

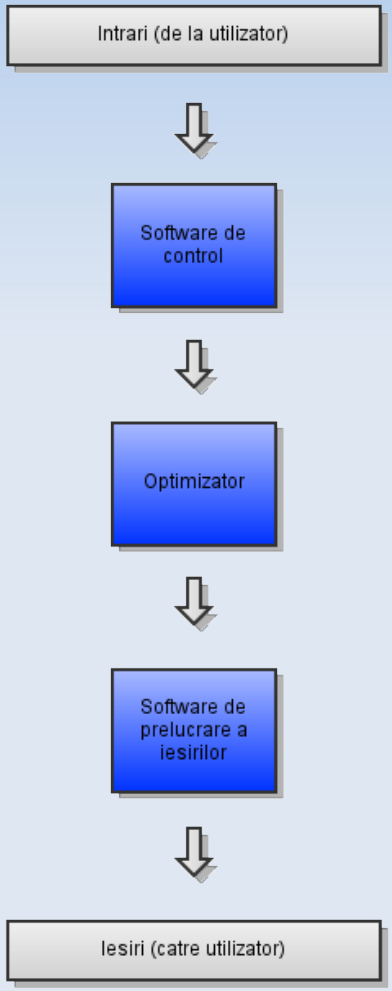
1. Problema. Formulare Matematica. Cautarea exhaustiva. Metode de Optimizare.
2. **Detalii de implementare ale sistemului de optimizare.**
3. Detalii algoritmi.
4. Concluzii.

2.Implementare

- Am realizat un sistem ce implementeaza algoritmii descrisi si rezolva problema gasirii retelei de acces optime.
- Pornind de la o descriere a problemei (un fisier de configurare ce codifica o configuratie de problema) obtinem o retea optima, conform metodei dorite.

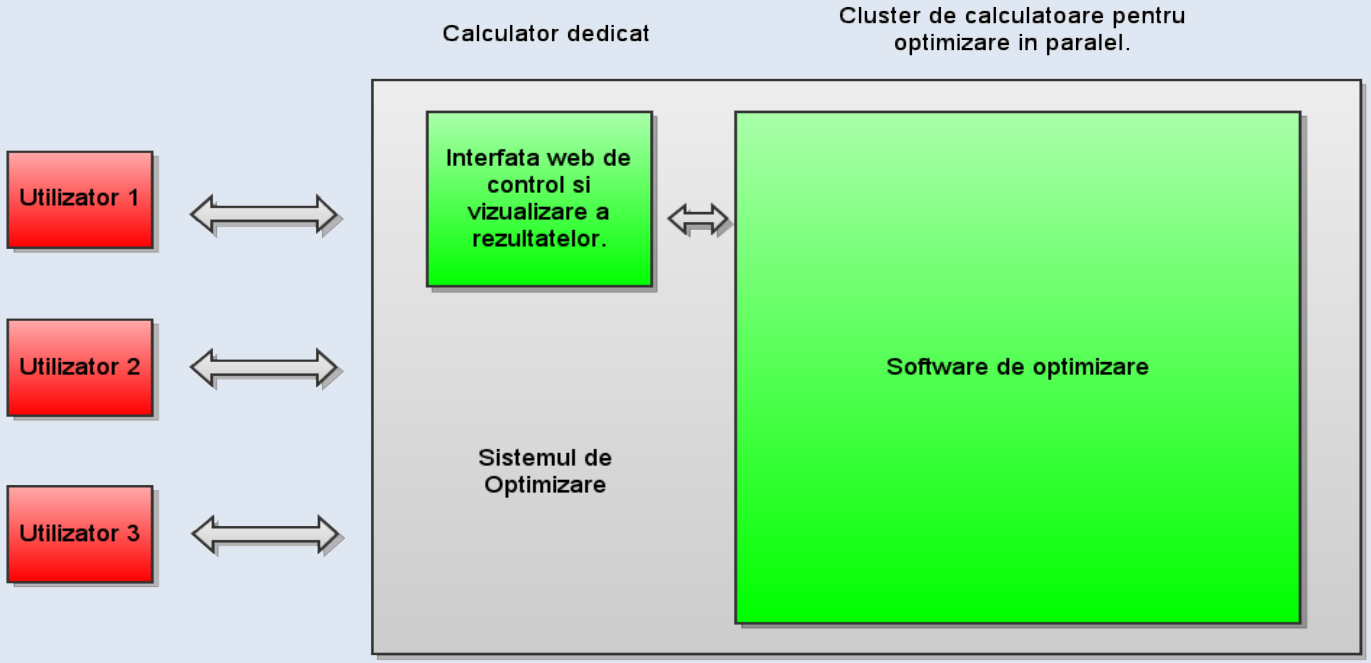
2.Implementare

- Structura sistemului (trei clase de programe) :
  - Optimizare.
  - Control.
  - Prelucrare a iesirilor.



2.Implementare

- Ideal, programul de control si prelucrare a iesirlor este o interfata web pentru intreg sistemul, iar optimizatorul ruleaza pe un sistem dedicat de diverse marimi.

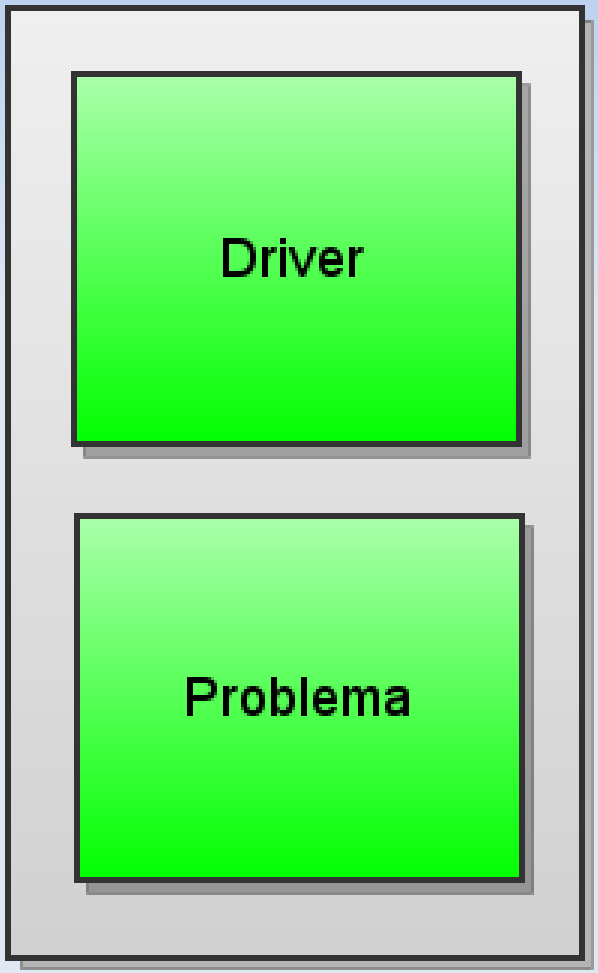


2.Implementare

- Programe de optimizare:
  - Cautare / BFS: Implementeaza cautari exhaustive pentru diverse probleme.
  - Optimizare / OPT: Implementeaza algoritmi de optimizare pentru diverse probleme.

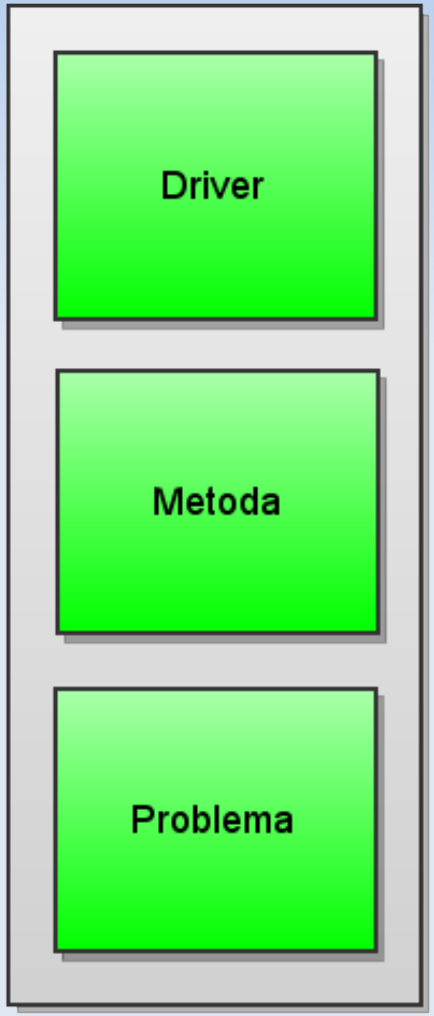
2.Implementare

- BFS: cautare exhaustiva.
- Folosit in testare si verificare.
- Structura:
  - Driver
  - Problema



2.Implementare

- OPT: metode de optimizare.
- Structura:
  - Driver
  - Metoda
  - Problema



2.Implementare

- **Problema**: defineste tot ce tine de descrierea reprezentarii unui individ sau de implementarea operatorilor genetici la nivel de individ (generare, mutatie, incrucisare).
- Trei exemple:
  - Retele de Acces
  - Functii reale 1-dimensionale
  - Functii reale 2-dimensionale

2.Implementare

- **Metoda**: defineste tot ce tine de descrierea reprezentarii unei populatii si a operatorilor genetici pe acestea (evolutie, selectie).
- Patru exemple:
  - Random Search
  - Hill Climbing
  - Evolution Strategy
  - Genetic Algorithm (8 variante).

2.Implementare

- **Driver**: defineste executia la nivel inalt a optimizatorului (conditia de stop, executie seriala/paralela, folosirea a mai multor calculatoare etc.)
- Un singur exemplu:
  - SingleThreadedDriver: executie seriala / clasica.

2.Implementare

- Module ortogonale: putem combina orice driver, cu orice metoda, cu orice problema.
- In total, avem 3 programe BFS si 36 programe OPT.
- Fiecare program face un singur lucru (optimizarea unei anume probleme) intr-un singur fel (cu o anume metoda si un anume driver).
- Programe derivate din surse comune.
- Implementare mai usoara.

2.Implementare

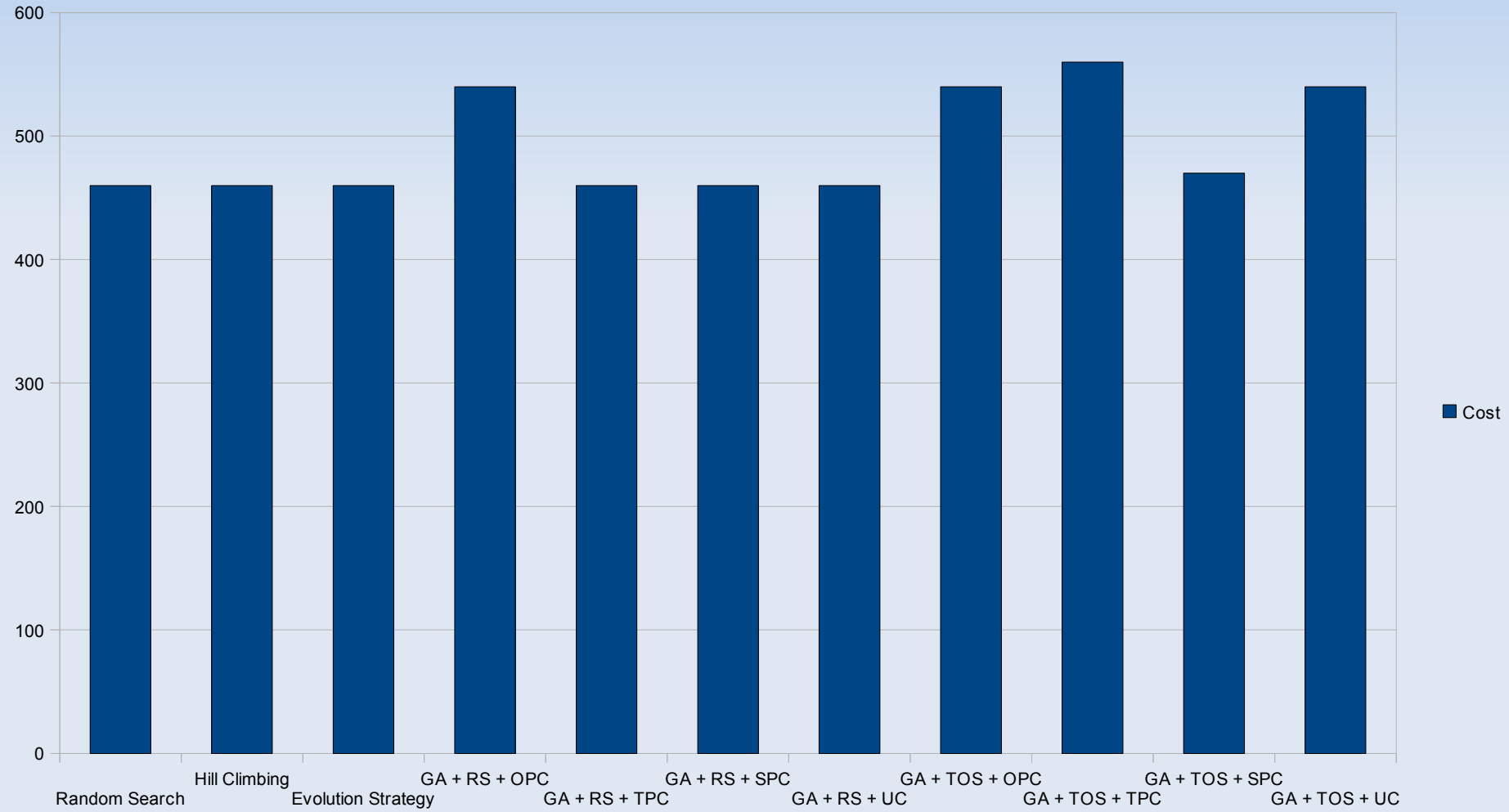
- Ca si mod de functionare, programele primesc o configuratie adecvata combinatiei de module ce le alcatuiesc si produc o descriere a procesului de evolutie.
- Intrarea poate fi produsa de un program de control, iar iesirea interpretata de un numar de programe de vizualizare.

2.Implementare

- Experimente pe doua instante de problema.
- Instanta 1:
  - 12 utilizatori.
  - 6 tipuri de noduri.
  - 100 iteratii / 10 indivizi in populatie.
- Instanta 2:
  - 1536 utilizatori.
  - 8 tipuri de noduri.
  - 100 iteratii / 2000 de indivizi in populatie.
  - ~1.3 minute timp de rulare

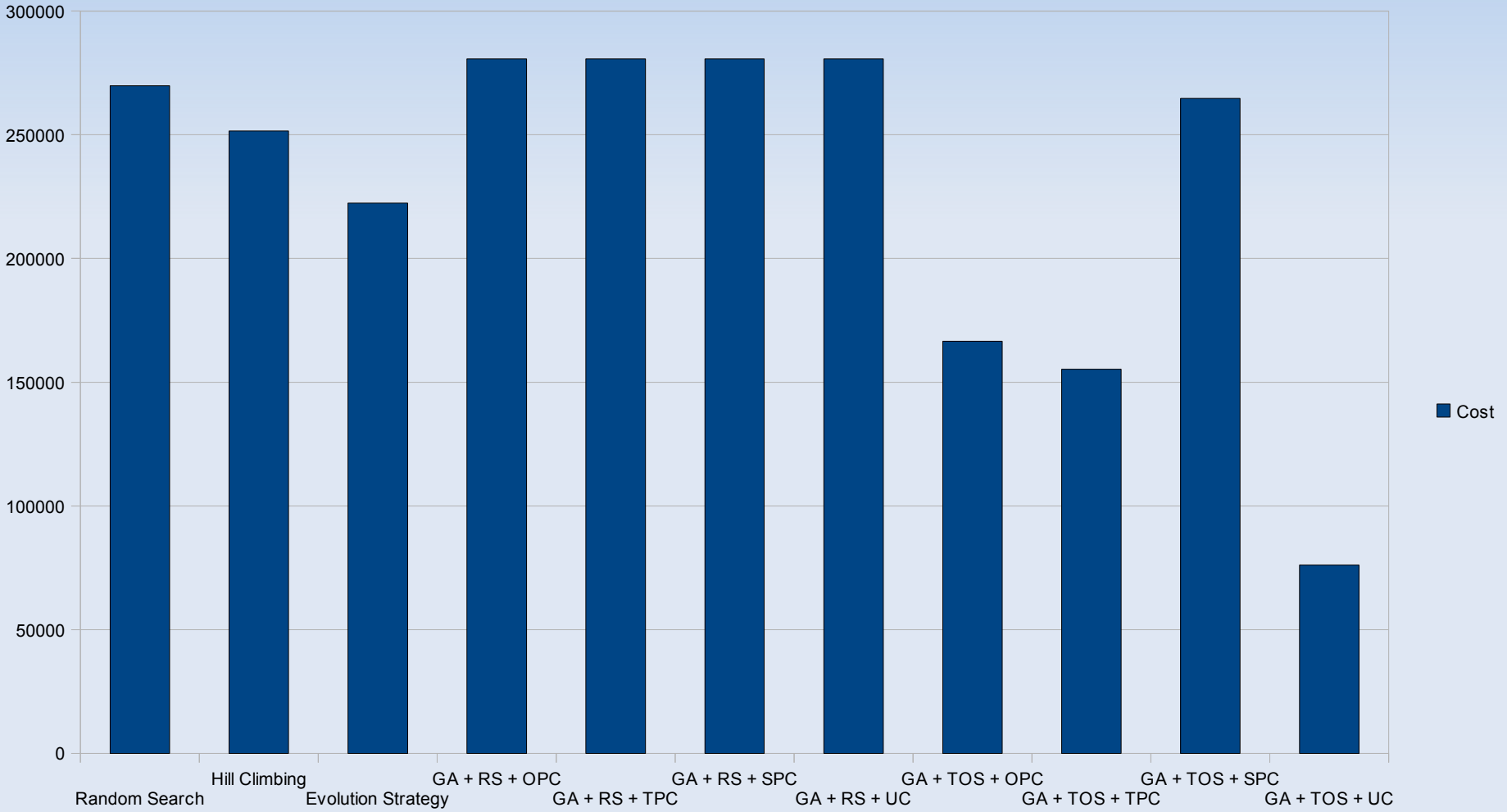
2.Implementare

Pe Instanta 1 avem urmatoarele rezultate (asemanatoare).



2.Implementare

Pe Instanta 2 avem urmatoarele rezultate (Genetic Algorithm fiind clar castigator).

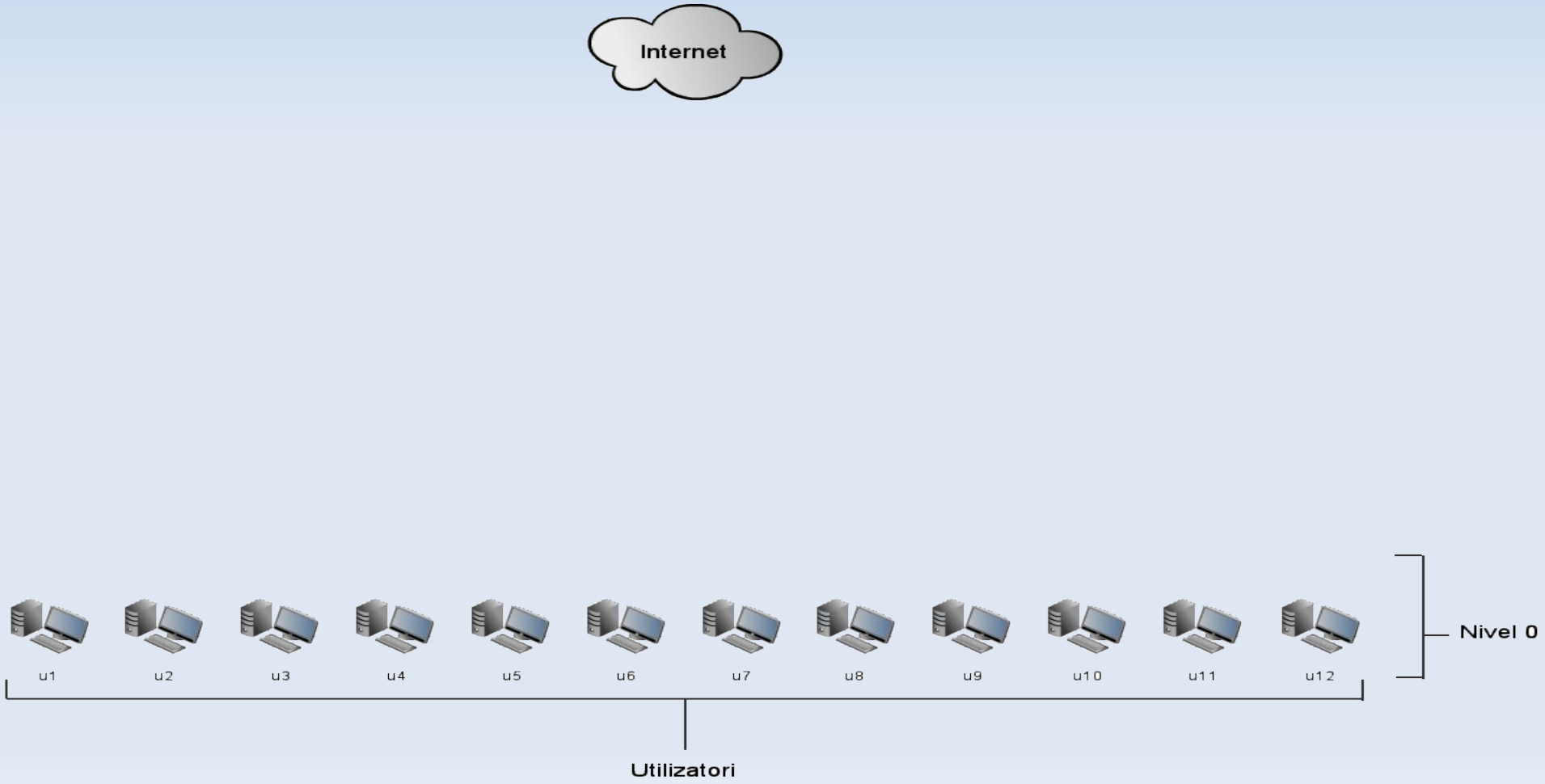


Capitolul 3

1. Problema. Formulare Matematica. Cautarea exhaustiva. Metode de Optimizare.
2. Detalii de implementare ale sistemului de optimizare.
3. **Detalii algoritmi.**
4. Concluzii.

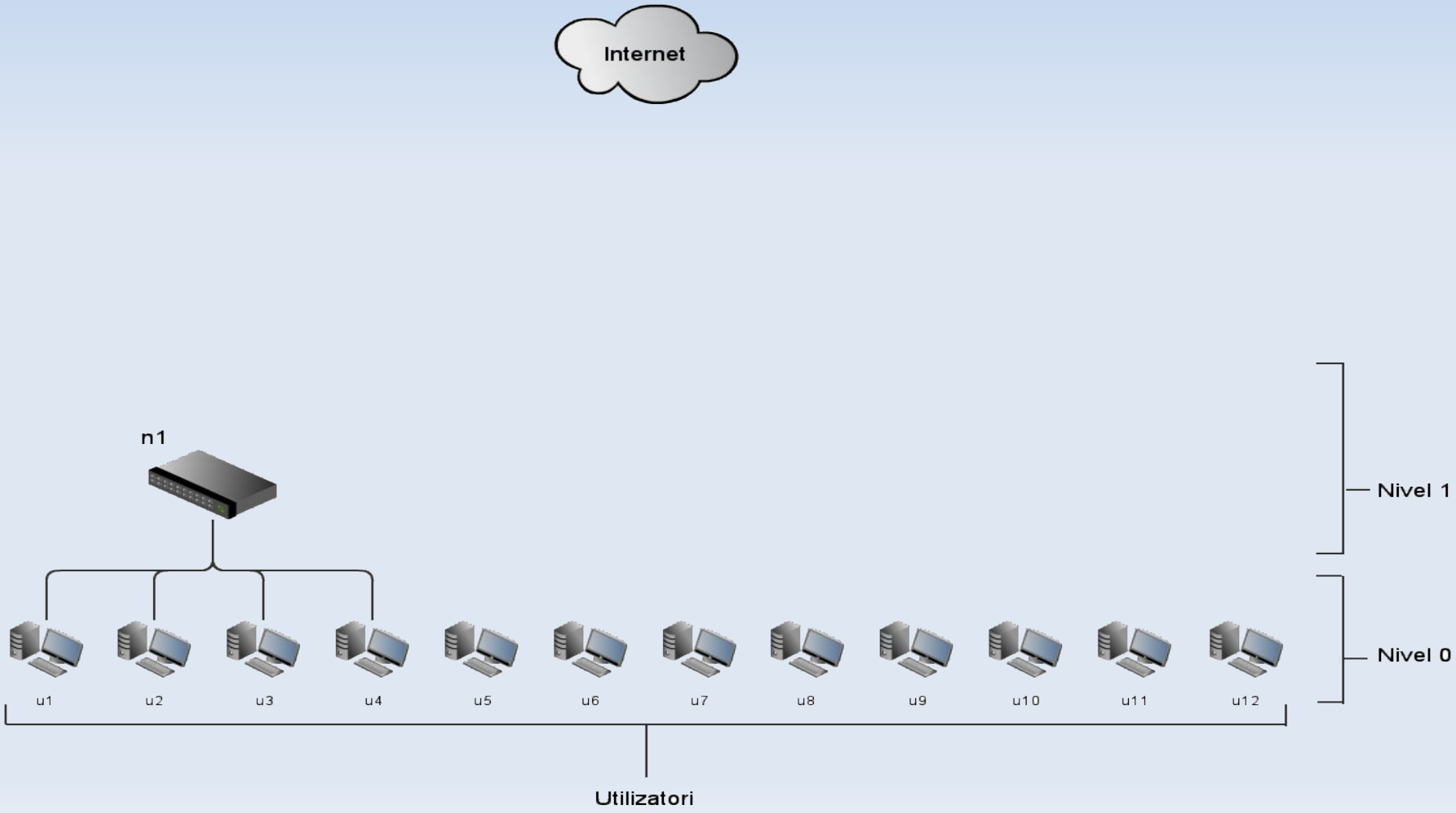
3.Algoritmi

- Evaluare descriere retea: (n1,n2,n1,n2,n4)



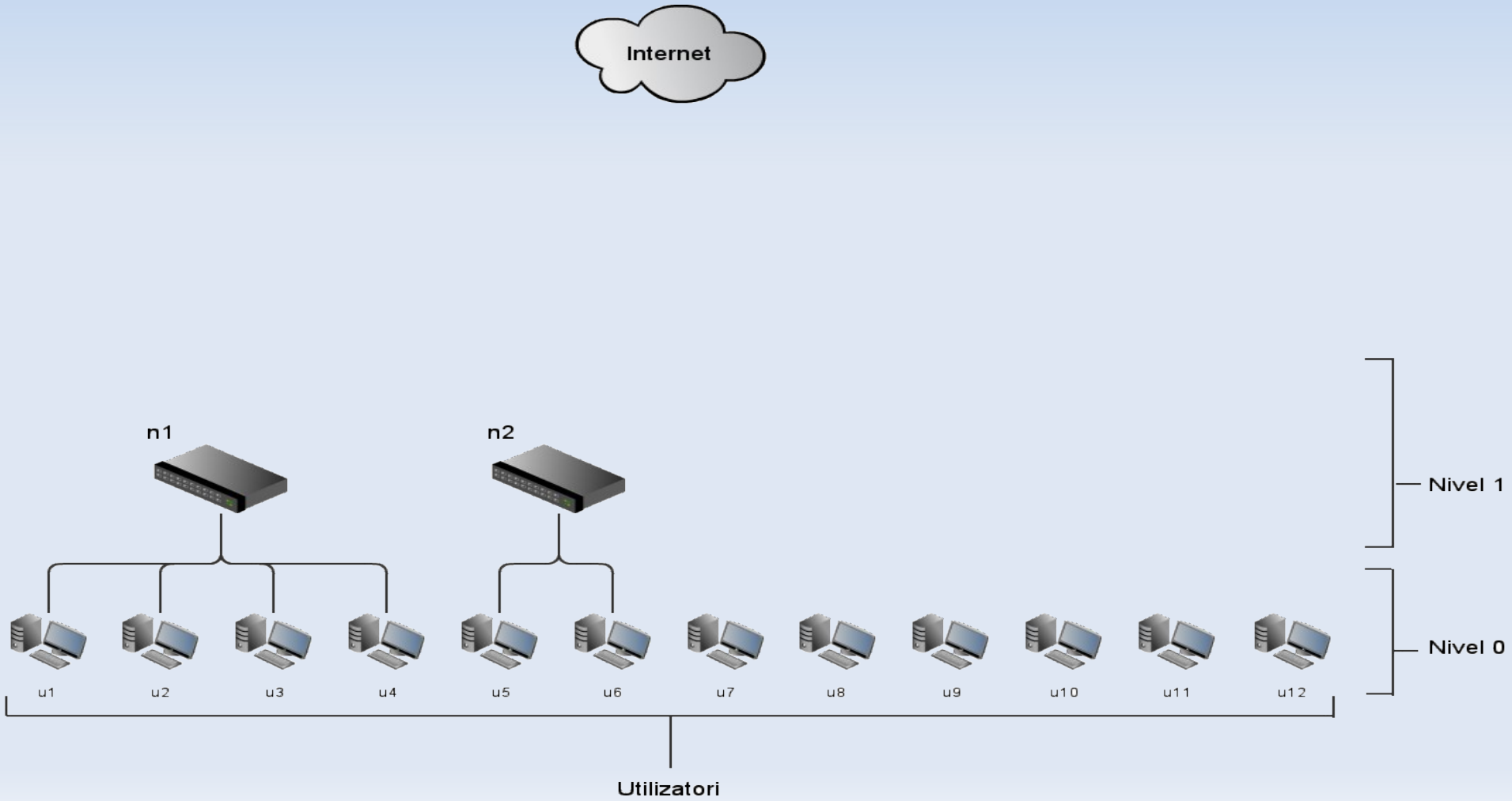
3.Algoritmi

- Evaluare descriere retea: (n1,n2,n1,n2,n4)



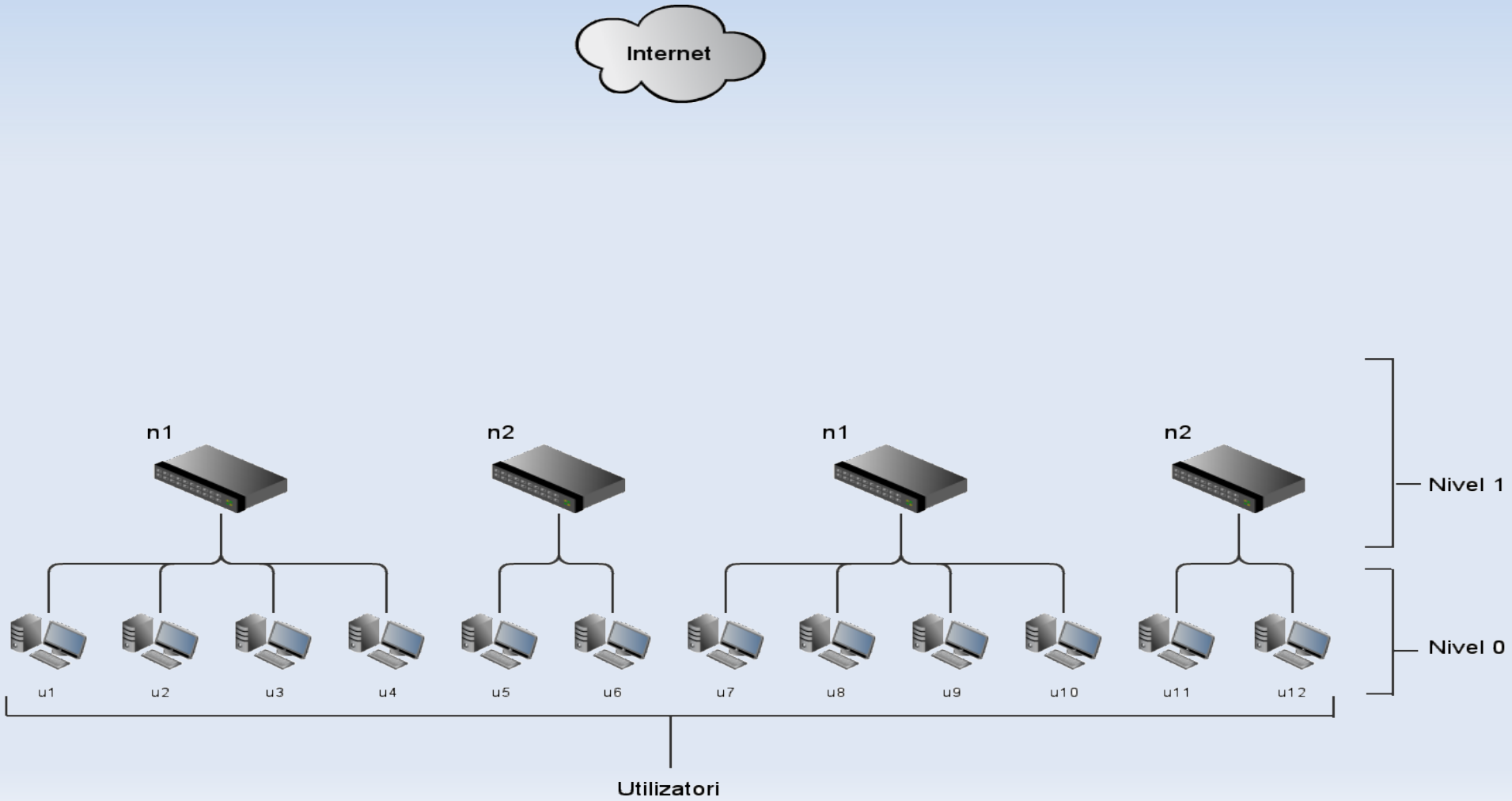
3.Algoritmi

- Evaluare descriere retea: (n1,n2,n1,n2,n4)



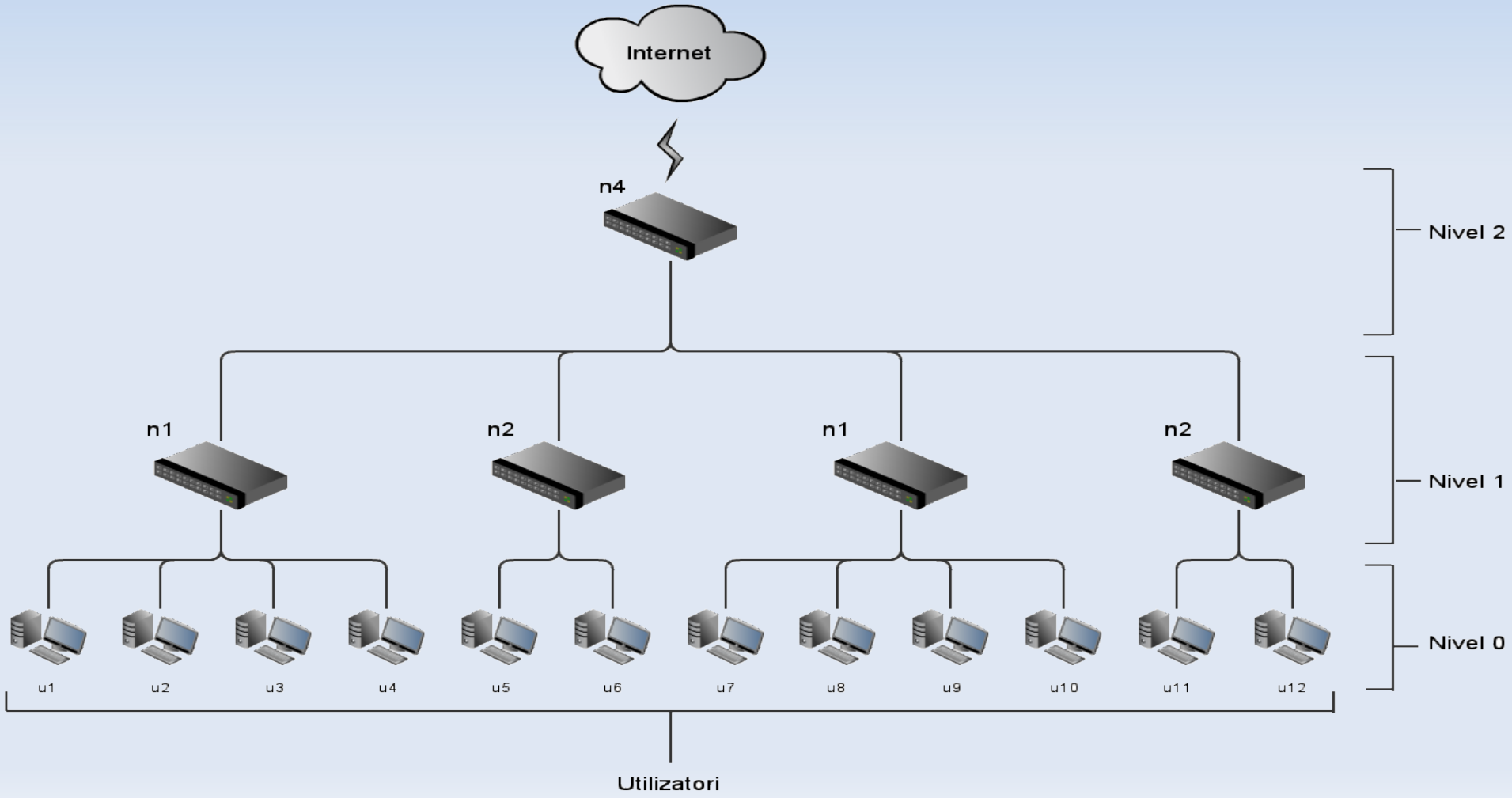
3.Algoritmi

- Evaluare descriere retea: (n1,n2,n1,n2,n4)



3.Algoritmi

- Evaluare descriere retea: (n1,n2,n1,n2,n4)



3.Algoritmi

Random Search

3.Algoritmi

Genetic Algorithm

Capitolul 4

1. Problema. Formulare Matematica. Cautarea exhaustiva. Metode de Optimizare.
2. Detalii de implementare ale sistemului de optimizare.
3. Detalii algoritmi.
4. **Concluzii.**

4.Concluzii

- Avem un sistem care merge, si poate gasii retele bune pentru configuratii mari.
- Directii viitoare:
  - Implementare interfata web pentru sistem.
  - Driver distribuit pentru optimizatoare.
  - Folosirea de informatii de legaturi in generarea retelei (pozitii ale utilizatorilor etc.)
  - Generearea de retele cu parte statica (pentru extinderea retelelor deja existente).