Problema satisfacerii restrictiilor

Curs 7

2010

Definirea PSR

- In *problema satisfacerii restrictiilor* – PSR (engl. *constraint satisfaction problem* – CSP), uneori numita si *problema selectiei multidimensionale* se dau:
  - O multime de variabile
  - Un domeniu (multime de valori posibile) pentru fiecare variabila
  - O multime de restrictii (sau constrangeri) si/sau o functie de evaluare
- In PSR se cere asignarea cate unei valori fiecarei variabile astfel incat:
  - Fie sunt satisfacute restrictiile, acestea numindu-se si *restrictii tari* (engl. *hard constraints*), iar problema numindu-se in acest caz *problema de satisfacere*.
  - Fie se minimizeaza functia de evaluare (sau cost), acest lucru numindu-se si *restrictie slaba* (engl. *soft constraint*), iar problema numindu-se in acest caz *problema de optimizare*.
- In numeroase probleme constrangerile tari apar amestecate cu constrangerile slabe.
- Intr-o PSR avem un numar finit de variabile, insa domeniile lor pot fi atat finite cat si infinite.

Legatura dintre PSR si cautare

- Exista cel putin 2 moduri de a privi o PSR ca o problema de cautare:
  - Un nod corespunde unei asignari de valori tuturor variabilelor problemei, iar vecinii unui nod corespund schimbarii unei valori a unei variabile. Un nod de start se obtine asignand aleator cate o valoare fiecarei variabile.
  - Se ordoneaza variabilele (V_1, V_2, …, V_n). Un nod corespunde unei asignari partiale de valori, doar primelor k variabile, $0 \leq k \leq n$. Vecinii unui nod se obtin prin asignarea unei valori variabilei urmatoare, cu indicele k+1. Nodul initial are toate variabilele neasignate, adica $k = 0$.
- Deosebire: in PSR nu ne intereseaza calea de la nodul initial catre un nod obiectiv, ci doar valoarea nodului obiectiv.
- In problemele de optimizare nodurile obiectiv nu sunt bine definite.
- In problemele de satisfacere nodurile obiectiv sunt definite implicit prin multimea de restrictii pe care trebuie sa la satisfaca.

Definirea formala a PSR

- O PSR se caracterizeaza prin:
  - Multimea variabilelor $\{V_1, V_2, ..., V_n\}$.
  - Pentru fiecare variabila V_i se indica domeniul sau D_i. D_i reprezinta multimea valorilor posibile ale variabilei V_i.
  - Pentru problemele de satisfacere se indica multimea restrictiilor. O restrictie reprezinta o submultime a produsului cartezian al tuturor domeniilor variabilelor.
  - O solutie a unei probleme de satisfacere reprezinta un *n*-tuplu de valori ale variabilelor care satisface toate restrictiile problemei.
  - Pentru problemele de optimizare se indica o functie care asociaza cate o valoare de cost fiecarei asignari de valori variabilelor problemei.
  - O solutie a unei probleme de optimizare reprezinta un *n*-tuplu de valori ale variabilelor care optimizeaza functia de cost.

Exemplu: planificarea activitatilor

- Sa presupunem ca robotul de distribuire trebuie sa planifice 5 activitati *a*, *b*, *c*, *d* si *e*, fiecare activitate poate incepe la unul din momentele 1, 2, 3 sau 4 iar durata unei activitati este neglijabila.
- Variabilele acestei probleme sunt momentele de realizare a activitatilor: *A*, *B*, *C*, *D* si *E*.
- Domeniile valorilor variabilelor problemei sunt $D_A = D_B = D_C = D_D = D_E = \{1,2,3,4\}$.
- Restrictiile problemei sunt:

  $(B \neq 3) \wedge (C \neq 2) \wedge (A \neq B) \wedge (B \neq C) \wedge (C < D) \wedge (A = D) \wedge (E < A) \wedge$

  $(E < B) \wedge (E < C) \wedge (E < D) \wedge (B \neq D)$
- Ele exprima urmatoarele lucruri: activitatea b nu poate fi realizata la momentul 3, *c* nu poate fi realizata la momentul 2, *a* nu poate fi realizata simultan cu *b*, *b* nu poate fi realizata simultan cu *c*, *c* trebuie realizata strict inaintea lui *d*, etc.

Reprezentarea PSR in Datalog

- O PSR in care domeniile variabilelor sunt multimi finite se numeste PSR finita – PSRF.
- O PSRF poate fi reprezentata in Datalog astfel:
  - Fiecare restrictie este o relatie *k*-ara si poate fi exprimata extensional prin multimea *k*-tuplelor sale. Fiecarui tuplu ii corespunde o clauza Datalog de forma unui fapt in care toti termenii sunt constante.
  - Multimea restrictiilor este reprezentata printr-o interogare Datalog in care variabilele corespund variabilelor problemei.
- Exemplu:

```
timp(1)   1 < 2   1 ≠ 2   3 ≠ 1   1 = 1
timp(2)   1 < 3   1 ≠ 3   3 ≠ 2   2 = 2
timp(3)   1 < 4   1 ≠ 4   3 ≠ 4   3 = 3
timp(4)   2 < 3   2 ≠ 1   4 ≠ 1   4 = 4
          2 < 4   2 ≠ 3   4 ≠ 2
          3 < 4   2 ≠ 4   4 ≠ 3

? timp(A) ∧ timp(B) ∧ timp(C) ∧ timp(D) ∧ timp(E) ∧
  (B ≠ 3) ∧ (C ≠ 2) ∧ (A ≠ B) ∧ (B ≠ C) ∧ (C < D) ∧
  (A = D) ∧ (E < A) ∧ (E < B) ∧ (E < C) ∧ (E < D) ∧
  (B ≠ D)
```

Rezolvarea PSRF prin generare si testare

- Consta in generarea sistematica prin cautare exhaustiva a tuturor elementelor produsului cartezian: $D_1 \times D_2 \times \ldots \times D_n$ si verificarea restrictiilor sau evaluarea functiei de cost pentru fiecare element in parte.
- Aceasta metoda necesita un timp exponential.
- Presupunand ca $|D_i| = d$ pentru toti $1 \leq i \leq n$, ca exista e restrictii care trebuie testate si ca testarea unei restrictii consuma un timp constant, rezulta ca algoritmul de generare si testare va necesita un timp de executie $O(ed^n)$.
- Exemplu: $D_A \times D_B \times D_C \times D_D \times D_E = \{(1,1,1,1,1), (1,1,1,1,2), ..., (4,4,4,4,4)\}$.
- Algoritmul de generare si testare se obtine aplicand procedura de demonstrare de sus in jos pentru interogarea Datalog si inlocuind nedeterminismul nu-conteaza cu selectia atomului cel mai din stanga din corpul clauzei raspuns (de ce ?)

Rezolvarea PSRF prin backtracking

- Asignarea unei valori unei variabile = instantiere.
- Presupune instantierea variabilelor intr-o anumita ordine si evaluarea fiecarei restrictii imediat ce variabilele din cadrul sau au devenit instantiate.
- Avantaj: orice instantiere partiala a variabilelor care nu satisface restrictiile va putea fi astfel rejectata.
- Spre exemplu orice instantiere partiala in care $A = 1$ si $B = 1$ va fi inconsistenta cu restrictia $A \neq B$.
- Astfel interogarea Datalog va putea fi modificata astfel incat, avand in vedere strategia de selectie de la stanga la dreapta, restrictiile sa fie impinse spre stanga astfel incat sa poata fi testate imediat ce variabilele corespunzatoare lor au devenit instantiate.
- Exemplu: `? timp(A) ∧ timp(B) ∧ (B ≠ 3) ∧ (A ≠ B) ∧ timp(C) ∧ (C ≠ 2) ∧ (B ≠ C) ∧ timp(D) ∧ (C < D) ∧ (A = D) ∧ (B ≠ D) ∧ timp(E) ∧ (E < A) ∧ (E < B) ∧ (E < C) ∧ (E < D)`

Graful de cautare pentru PSRF

- PSRF poate fi privita ca problema de cautare intr-un graf astfel:
  - Se ordoneaza variabilele V_1, V_2, ..., V_n.
  - Fiecarui nod ii corespunde o instantiere a primelor j variabile. Aceasta instantiere poate fi descrisa prin substitutia: $\{V_1/v_1, ..., V_j/v_j\}$.
  - Vecinii unui nod $\{V_1/v_1, ..., V_j/v_j\}$ sunt nodurile de forma $\{V_1/v_1, ..., V_j/v_j, V_{j+1}/v_{j+1}\}$ cu $v_{j+1} \in D_{j+1}$ care sunt consistente cu multimea de restrictii.
  - Nodul de start este asignarea vida $\{\}$.
  - Un nod obiectiv este o instantiere a tuturor variabilelor care este consistenta cu multimea restrictiilor problemei.
- Algoritmul backtracking corespunde unei cautari in adancime a acestui graf.
- Eficienta algoritmului backtracking depinde de modul de ordonare a vaiabilelor problemei.

Algoritmi de consistenta

- Ideea de baza este de a elimina din domeniile de valori ale variabilelor acele valori care nu pot fi luate in considerare.
- Spre exemplu asignarea $C = 4$ este inconsistenta cu celelalte asignari posibile deoarece $D_C = D_D = \{1,2,3,4\}$ si $C < D$. Algoritmul backtracking va redescoperi acest lucru de mai multe ori, find astfel ineficient. Cel mai simplu este sa se elimine valoarea 4 din D_C de la inceput.
- Definitie. O variabila se numeste *consistenta in raport cu domeniul sau* (engl. *domain consistent*) daca pentru orice valoare din domeniu nu exista nici o restrictie care sa fie inconsistenta cu aceasta valoare.
- In exemplul de mai sus variabila C nu este consistenta in raport cu domeniul $D_C = \{1,2,3,4\}$. Analog, pentru problema de planificare a activitatilor variabila B nu este consistenta cu $D_B = \{1,2,3,4\}$ deoarece exista restrictia $B \neq 3$ care este inconsistenta cu valoarea $3 \in D_B$.

Arc consistenta

- O PSR se numeste *k*-ara daca toate restrictiile sale au aritatea cel mult *k*. O PSR *k*-ara cu $k = 2$ se numeste PSR binara – PSRB.
- O PSRB se poate reprezenta printr-o *retea de restrictii* (engl. *constraint network*). Fiecare nod al retelei corespunde unei variabile si fiecare pereche de arce (X,Y) si (Y,X) intre nodurile *X* si *Y* corespunde unei restrictii $P(X,Y)$.
- Un arc (X,Y) se numeste *arc consistent* daca pentru orice valoare a variabilei *X* din D_X exista o valoare a variabilei *Y* din D_Y astfel incat relatia $P(X,Y)$ este satisfacuta.
- O retea de restrictii se numeste *arc consistenta* daca toate arcele sale sunt arc consistente. O retea de restrictii se numeste *consistenta in raport cu domeniul* daca toate nodurile sale sunt consistente in raport cu domeniul.
- Daca un arc (X,Y) nu este arc consistent atunci toate valorile lui *X* din D_X pentru care nu exista nici o valoare *Y* in D_Y astfel incat $P(X,Y)$ este satisfacuta pot fi eliminate.

Algoritmul AC-3

Intrare:

Multimea variabilelor

Domeniul D_X pentru fiecare variabila X

Restrictiile unare P_X pentru fiecare variabila X

Restrictiile binare P_{XY} pentru fiecare pereche de variabile X si Y

Iesire: o retea domeniu si arc consistenta

Pentru fiecare variabila X

$D_X \leftarrow \{x \in D_X \mid P_X(x)\}$

$TDA \leftarrow \{(X,Y) \mid P_{XY}$ este o constrangere binara$\} \cup$

$\{(Y,X) \mid P_{XY}$ este o constrangere binara$\}$

Repeta

Selecteaza un arc $(X,Y) \in TDA$

$TDA \leftarrow TDA \setminus \{(X,Y)\}$

$ND_X \leftarrow D_X \setminus \{x \in D_X \mid$ nu exista nici un $y \in D_Y$ astfel incat $P_{XY}(x,y)\}$

Daca $ND_X \neq D_X$ atunci

$TDA \leftarrow TDA \cup \{(Z,X) \mid Z \neq Y\}$

Pana cand TDA = $\varnothing$

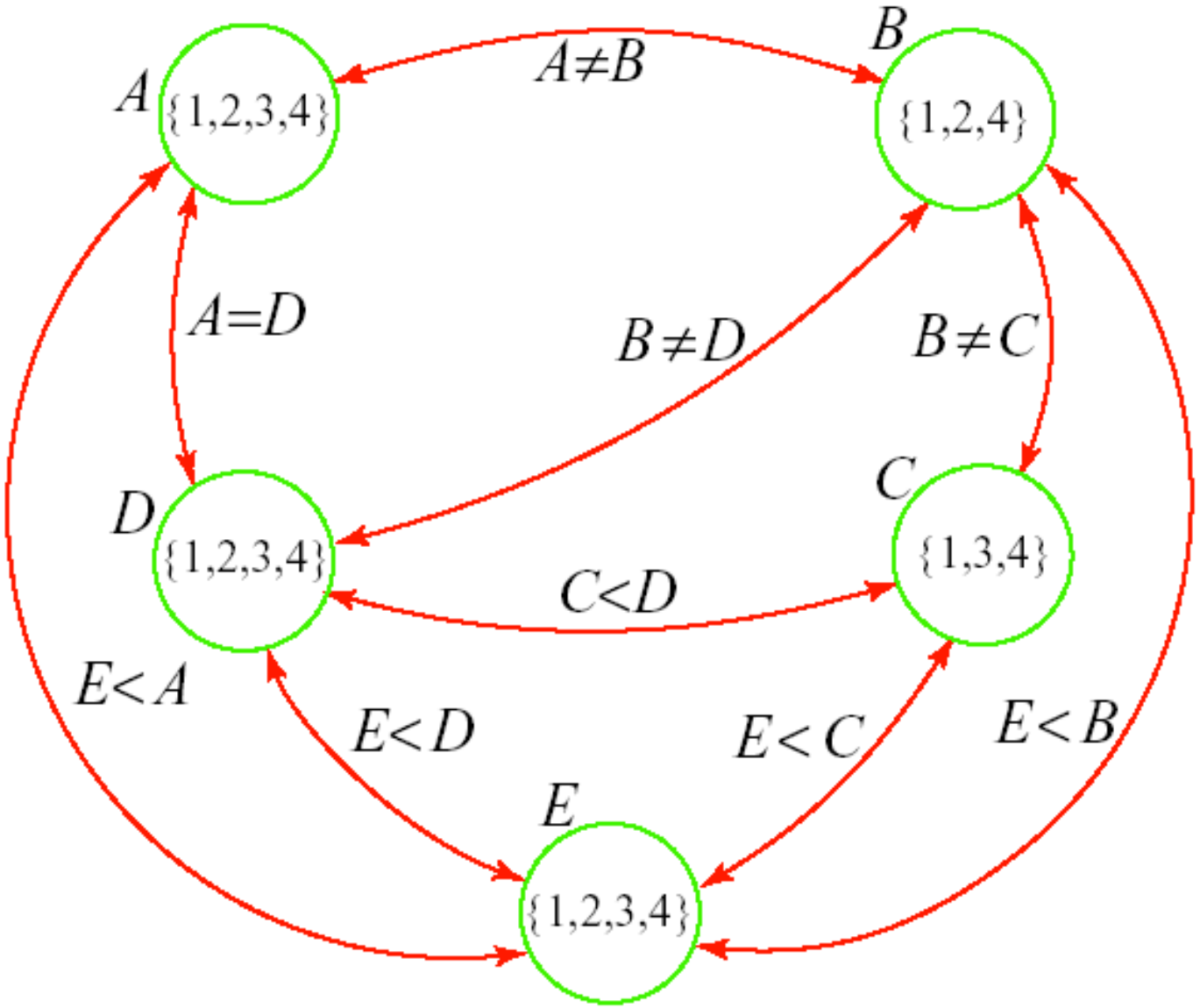
Complexitatea algoritmului AC-3

Time complexity = $O(e \times d^3)$

e = number of constraints (binary)

$d = max_{\ 1 \le i \le n} |D_i|$

Exemplu de retea de constrangeri



Rezolvarea PSR cu algoritmul AC-3

- In urma executiei algoritmului AC-3 pentru o retea de restrictii date rezulta o retea de restrictii arc si domeniu consistenta. Ea este aceeasi indiferent de ordinea in care sunt verificate arcele.
- La terminarea algoritmului AC-3 avem 3 posibilitati:
  - Toate domeniile sunt vide, caz in care problema nu are solutii.
  - Toate domeniile au un singur element, caz in care problema are solutie unica.
  - Exista cel putin un domeniu cu mai mult de un element. In acest caz se alege un astfel de domeniu, se injumatateste si algoritmul AC-3 se aplica recursiv problemelor PSR rezultate.

Metoda urcarii dealului

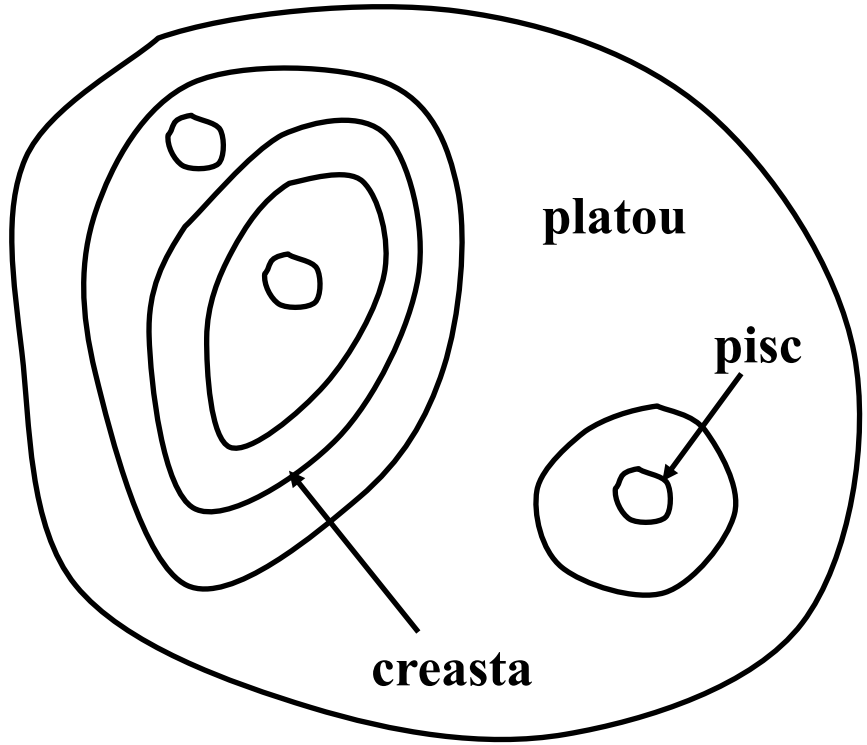
- O metoda utila in practica pentru rezolvarea PSR de satisfacere si optimizare, pentru cazul in care spatiul starilor este foarte mare sau infinit este metoda urcarii dealului (engl. *hill climbing*).
  - Necesita existenta unei functii euristice care asociaza o valoare fiecarei instantieri complete a variabilelor.
  - Fiecare nod al grafului de cautare contine o instantiere completa a variabilelor problemei.
  - La fiecare pas se selecteaza un "vecin" al instantierii curente care o imbunatateste.
- O astfel de metoda se numeste si cautare prin *imbunatatire iterativa* (engl. *iterative improvement*).

Selectarea vecinilor

- Daca domeniile sunt neordonate atunci vecinii unui nod se construiesc prin alegerea unei alte valori pentru una din variabilele problemei. Daca avem n variabile si fiecare domeniu are m valori atunci vom avea $n(m\text{-}1)$ posibilitati.
- Daca domeniile sunt ordonate atunci vecinii se obtin considerand valorile adiacente pentru una dintre variabile. Daca problema are n variabile atunci vom avea $2n$ vecini.
- Daca domeniile sunt continue atunci se poate utiliza gradientul functiei de optimizare. Se alege un pas p si se determina un vecin $V = w$ al instantierii curente $V = v$ cu formula $w = v + p(dh/dV)$. Daca $p > 0$ atunci se determina un maxim al lui h. Daca $p < 0$ atunci se determina un minim al lui h. Metoda corespunzatoare de optimizare se numeste *metoda gradientului*.

Probleme ale metodei urcarii dealului

- *Piscuri* (engl. *foothill*) = extreme locale
- *Platouri* (engl. *plateaux*) = noduri in care vecinii au aceeasi valoare ca si nodul curent.
- *Creste* (engl. *ridge*) = noduri in care vecinii au o valoare mai mica decat nodul curent, dar facand o combinatie de pasi se poate obtine o imbunatatire. De exemplu vecinii pot sa Indice o deplasare inspre nord, sud, est si vest iar imbunatatirea sa se faca doar prin deplasare spre nord-est.



Imbunatatiri ale metodei urcarii dealului

- Combinarea cu metoda backtracking: se efectueaza reveniri atunci cand se atinge un extrem local.
- Repornirea metodei urcarii dealului de la un nod initial generat aleator.
- Alegerea aleatoare a nodului vecin si raportarea la sfarsit a valorii extreme gasite.
- Ultimele doua metode se pot combina: se face intai o cautare cu selectia aleatoare a vecinului si se pastreaza nodul cu valoarea maxima, si apoi de la acest nod se continua cu urcarea dealului, varianta normala.
- O alta varianta este de a nu selecta intodeauna la fiecare pas varianta cea mai buna, ci din cand in cand sa se selecteze si un nod care nu este cel mai bun. Aceasta metoda este cunoscuta in literatura sub numele de metoda calirii simulate (engl. *simulated annealing*). Este inspirata de tehnologia de calire a metalelor din metalurgie.

Metoda calirii simulate

- Alegerea din cand in cand a nodului mai slab se face cu o probabilitate care scade in raport cu cresterea numarului de iteratii.
- Probabilitatea depinde de o functie de temperatura care creste odata cu cresterea numarului de iteratii.

$n \leftarrow$ nod initial generat aleator

$T \leftarrow$ valoarea initiala a functiei de temperatura

Repeta

  Selecteaza aleator un vecin n' al nodului curent n

  Daca $h(n') \geq h(n)$ atunci

    $n \leftarrow n'$

  Altfel

    $n \leftarrow n'$ cu probabilitatea $exp((h(n)-h(n'))/T)$

  Se reduce valoarea lui T

Pana cand se atinge o conditie de oprire

Cautare cu fascicol si algoritmi genetici

- *Cautarea cu fascicol* (engl. *beam search*) este o imbunatatire a metodei urcarii dealului. In loc de a retine la fiecare iteratie un singur nod cel mai bun, se retin cele mai bune $k \geq 1$ noduri. Aceasta multime de noduri se numeste *populatie*.
- Pentru $k = 1$ cautarea cu fascicol se reduce la urcarea dealului si pentru $k = \infty$ se reduce la cautarea pe nivel.
- Cautarea cu fascicol este utila atunci cand exista restrictii de memorie, putandu-se alege in acest caz o valoare convenabila pentru k in functie de volumul de memorie disponibila.
- *Cautarea aleatoare cu fascicol* selecteaza aleator cei k vecini ai nodului curent. Se pot folosi probabilitati a.i. probabilitatea alegerii unui nod mai bun este mai mare.
- *Algoritmii genetici* sunt asemanatori cu cautarea aleatoare cu fascicol. Diferenta este in modul de generare a urmatoarei populatii. Sunt inspirati din genetica si de ideea ca indivizii care supravietuiesc sunt cei mai buni.
- Algoritmii genetici folosesc de regula trei operatori:
  - Selectie – se selecteaza pentru imperechere nodurile cele mai bune. Acestea vor participa la operatia de incrucisare.
  - Incrucisare (engl. *crossover*) – din doua noduri se genereaza un nod nou.
  - Mutatie – se modifica aleator un nod rezultand un nod mutant.