Algoritmi de căutare aleatoare. Simulated Annealing

- Motivație
- Algoritmi simpli de căutare aleatoare
- Algoritmi de tip Simulated Annealing

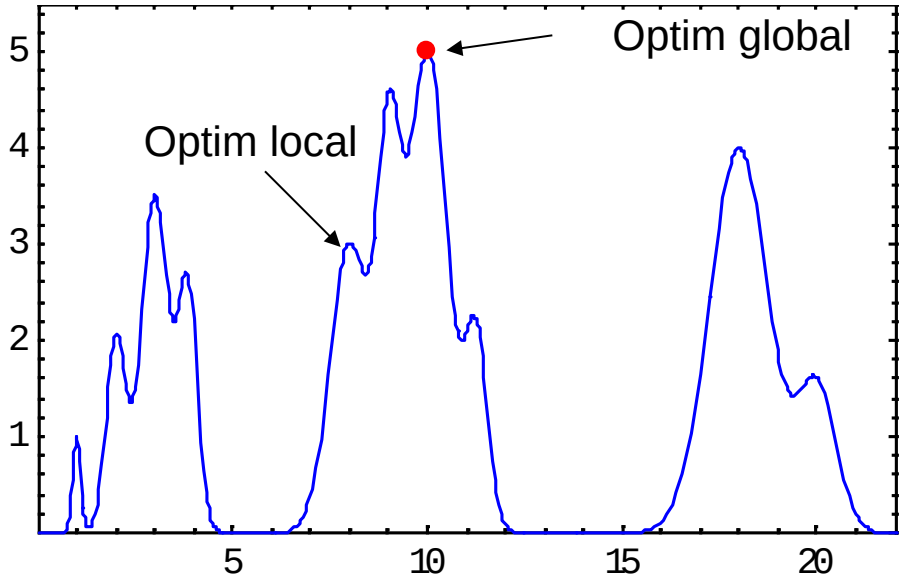
Motivație

Optimizare globală:

- identificarea optimului global al unei funcții: pentru o problemă de maximizare se caută x* cu proprietatea că f(x*)>=f(x), pentru orice x
- daca funcția obiectiv are și optime locale atunci metodele de căutare locală (cum este metoda gradientului) se pot bloca în punctele de optim local

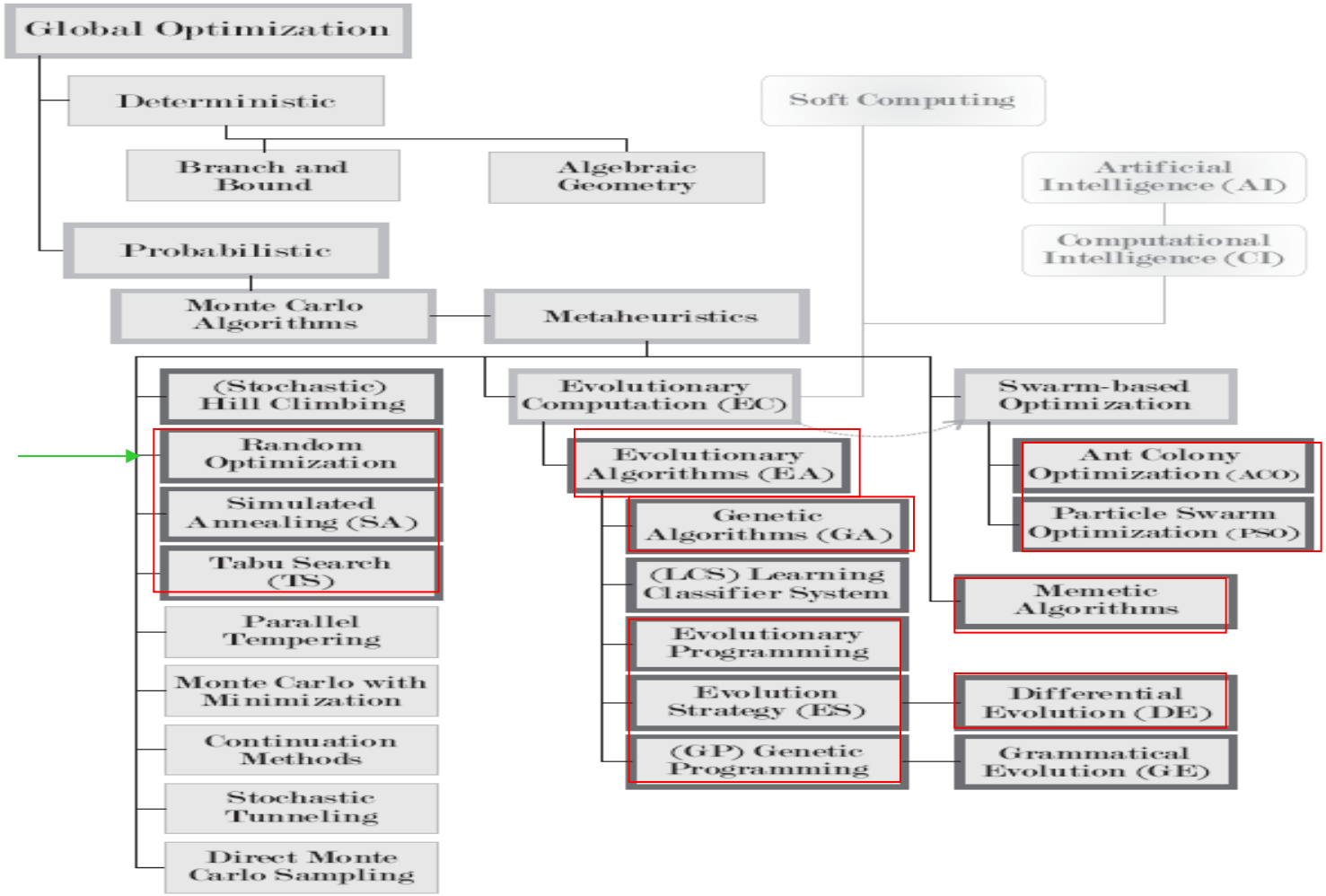
Exemplu:

- Antrenarea unei rețele neuronale



Motivație

Taxonomia algoritmilor de optimizare globală



Algoritmi de căutare aleatoare

Introducerea unor elemente aleatoare intr-un proces de căutare a soluției unei probleme de optimizare permite evitarea blocării in optime locale și apropierea de optimul global

Exemplu:

Inlocuirea direcției dată de gradient cu o direcție aleatoare

Avantaj: simplitate, nu necesită satisfacerea de către funcția de optimizat a unor proprietăți de netezime; este suficient să se poată evalua funcția – util în special când funcția obiectiv poate fi evaluată doar numeric (ex: simularea comportării unui sistem)

Dezavantaj: nu garantează convergența către soluție (în majoritatea cazurilor se poate demonstra doar o convergență în sens probabilist)

Algoritmi de căutare aleatoare

Idee:

- aproximația curentă este perturbată aleator
- daca prin perturbare se ajunge la o configurație mai bună se acceptă această perturbare

Problema: se cauta x* care minimizează o funcție f

Structura generală:

Initializare x

Repeat

z := generare perturbare aleatoare

If f(x+z)<f(x) then x:=x+z

Until " este satisfacută o condiție de oprire"

Exemplu: algoritmul Matyas (1960)

Init x(0)

k:=0 // contor iteratie

e:=0 // numar cazuri de esec

REPEAT

  generează un vector aleator cu componente normal repartizate $(z_1,\ldots z_n)$

  IF f(x(k)+z)<f(x(k)) THEN x(k+1):=x(k)+z

  e:=0

  ELSE x(k+1):=x(k)

  e:=e+1

  k:=k+1

UNTIL (k=kmax) OR (e=emax)

Obs. Perturbația aleatoare se aplică de regulă asupra unei singure componente (vectorul z are o singură componentă nenulă)

Problema: cum se aleg parametrii repartiției folosite pentru perturbarea valorii curente ?

Exemplu: N(0,s)

Exemplu: algoritmul Matyas (1960)

Pentru generarea valorilor aleatoare cu repartiție normală (N(0,1)) se poate folosi algoritmul Box-Muller

u:=random(0,1) // valoare aleatoare uniform repartizată in (0,1)

v:=random(0,1)

r:=sqrt(-2*ln(u));

z1:=r*cos(2*PI*v)

z2:=r*sin(2*PI*v)

RETURN z1,z2 // valorile z1 si z2 pot fi considerate independente

Exemplu: algoritmul Solis-Wets (1981)

Init x(0)

k:=0; m(0):=0 // media vectorului perturbatie este ajustata in timp

REPEAT

generează un vector aleator (z_1,…z_n) cu componente avand repartitia N(m(k),1)

IF f(x(k)+z)<f(x(k)) THEN x(k+1):=x(k)+z; m(k+1):=0.4*z+0.2*m(k)

IF f(x(k)-z)<f(x(k)) THEN x(k+1):=x(k)-z; m(k+1):=m(k)-0.4*z

IF f(x(k)-z)>f(x(k)) AND f(x(k)+z)>f(x(k)) THEN

x(k+1):=x(k)

m(k+1):=0.5*m(k)

k:=k+1

UNTIL (k=kmax)

Aplicații

Rețele neuronale: algoritmii aleatori de tip Matyas sau Solis-Wets au fost aplicați în antrenarea rețelelor neuronale feed-forward

Idee: se inlocuiește algoritmul BackPropagation cu un algoritm aleator

Efect: rezolvă parțial problema blocării în minime locale

Obs:

1. comportarea unui astfel de algoritm se evaluează prin execuția sa de mai multe ori și realizarea unui studiu statistic

2. Algoritmii de acest tip fac parte din clasa algoritmilor de tip Monte-Carlo

Simulated Annealing

Idee:

- se acceptă, cu o anumita probabilitate, și ajustari ale configurației curente care conduc la creșterea funcției obiectiv

Sursa de inspirație:

- procesul de reorganizare a structurii unui solid supus unui tratament termic:
  - Solidul este încălzit (topit): particulele sunt distribuite într-o manieră aleatoare
  - Solidul este răcit lent: particulele se reorganizează pentru a se ajunge la configurații de energie din ce în ce mică

Terminologie:

simulated annealing = tratament termic simulat = călire simulată

Istoric: Metropolis(1953), Kirkpatrick, Gelatt, Vecchi (1983), Cerny (1985)

Simulated Annealing

Analogie:

Proces fizic:		Problema de minimizare:
• Energia sistemului	→	Funcție obiectiv
• Starea sistemului	→	Configuratie (soluție candidat)
• Modificarea stării sistemului	→	Perturbarea configurației curente
• Temperatura	→	Parametru de control a procesului de optimizare

SA= metoda euristică inspirată de procese fizice

Simulated Annealing

Puțină fizică:

- fiecare stare a unui sistem este caracterizată de o anumită probabilitate de apariție

- probabilitatea asociată unei stări depinde de energia stării și de temperatura sistemului (ex: distribuția Boltzmann)

$$P_T(s) = \frac{1}{Z(T)} \exp(-\frac{E(s)}{k_B T})$$

$$Z(T) = \sum_{s \in S} \exp(-\frac{E(s)}{k_B T})$$

E(s) = energia stării s
T = temperatura sistemului
k_B = constanta Boltzmann
Z(T)=funcția de partiție
(factor de normalizare)

Simulated Annealing

Puțină fizică:

- Valori mari ale lui T (T tinde la infinit): argumentul lui exp este aproape 0 => stările sunt echiprobabile

- Valori mici ale lui T (T tinde la 0): vor avea probabilitate nenulă doar stările cu energia nulă

$$P_T(s) = \frac{1}{Z(T)} \exp(-\frac{E(s)}{k_B T})$$

$$Z(T) = \sum_{s \in S} \exp(-\frac{E(s)}{k_B T})$$

E(s) = energia stării s
T = temperatura sistemului
k_B = constanta Boltzmann
Z(T)=funcția de partiție
(factor de normalizare)

Simulated Annealing

Cum folosim acest lucru pentru rezolvarea unei probleme de optimizare ?

- Ar fi suficient să generăm configurații în conformitate cu distribuția Boltzmann pentru valori din ce în ce mai mici ale temperaturii

- **Problema:** dificil de calculat Z(T) (presupune calculul unei sume pentru toate stările posibile, adică generarea tuturor configurațiilor spațiului de căutare - IMPOSIBIL de realizat practic dacă spațiul configurațiilor este mare)

- **Solutie:** se aproximează distribuția prin simularea evoluției unui proces stohastic (lanț Markov) a cărui distribuție staționară coincide cu distribuția Boltzmann => algoritmul Metropolis

Simulated Annealing

Algoritmul Metropolis (1953)

Init x(0)

k:=0

REPEAT

x':=perturb(x(k))

IF f(x')<f(x(k)) THEN x(k+1):=x' (necondiționat)

ELSE x(k+1):=x'

cu probabilitatea min{1,exp(-(f(x')-f(x(k))/T)}

k:=k+1

UNTIL "este indeplinita o conditie de terminare"

Simulated Annealing

Algoritmul Metropolis – proprietăți

- Alta probabilitate de acceptare:

  $P(x(k+1)=x') = 1/(1+\exp((f(x')-f(x(k))/T))$

- Implementarea unei reguli de atribuire cu o anumită probabilitate

  u:=Random(0,1)

  IF u<P(x(k+1)=x') THEN x(k+1)=x'

  ELSE x(k+1)=x(k)

- Valori mari pentru T -> probabilitate mare de acceptare a oricarei configurații (similar cu căutarea pur aleatoare)

  Valori mici pentru T -> probabilitate mare de acceptare doar pentru configurațiile care conduc la micșorarea funcției obiectiv (similar cu o metodă de descreștere sau căutare de tip greedy)

Simulated Annealing

Algoritmul Metropolis – proprietăți

- Generarea unor noi configurații depinde de problema de rezolvat

Optimizare în domenii continue

$x'=x+z$

$z=(z_1,\ldots,z_n)$

z_i : generata in cf. cu repartitia

- N(0,T)
- Cauchy(T) (Fast SA)
- altele

Optimizare combinatorială

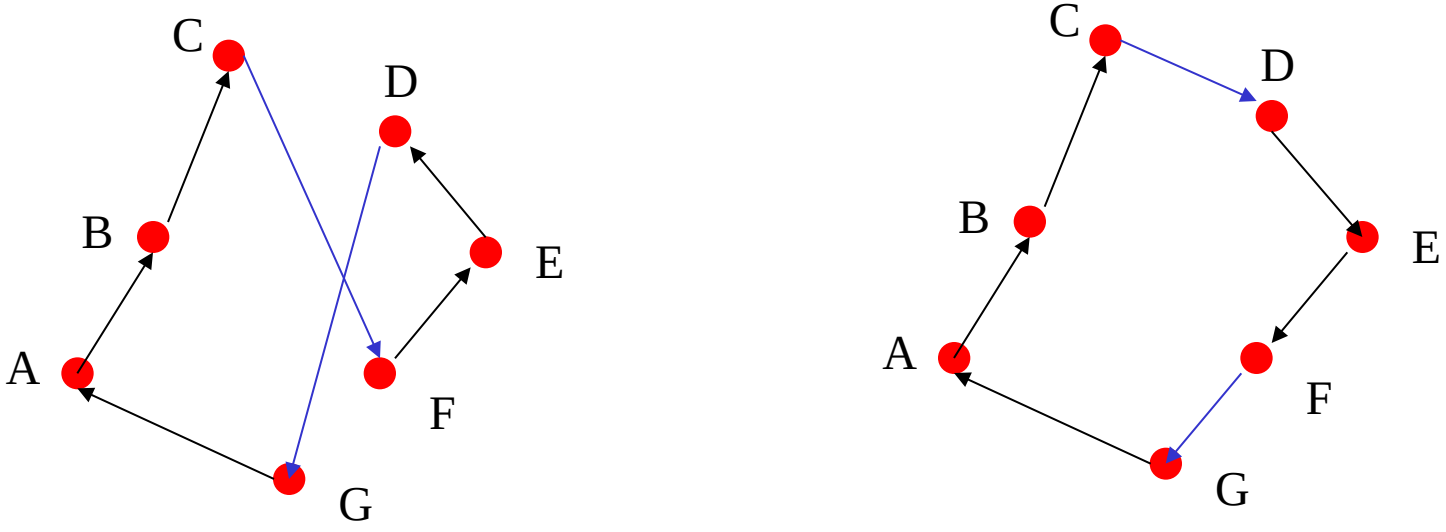
Noua configurație se selectează determinist/aleator din vecinătatea configurației curente

Exemplu: TSP – transformare de tip 2-opt

Simulated Annealing

TSP (Travelling Salesman Problem)

- Generarea unei noi configurații (transformare 2-opt)



ABCFEDG → ABCFEDG → ABCDEFG

Simulated Annealing

Simulated Annealing = aplicare repetată a algoritmului Metropolis pentru valori din ce în ce mai mici ale temperaturii

Structura generală

Init x(0), T(0)

i:=0

REPEAT

  aplică Metropolis (pentru una sau mai multe iteratii)

  calcul T(i+1)

  i:=i+1

UNTIL T(i)<eps

Problema: alegerea schemei de modificare a temperaturii (“cooling scheme”)

Simulated Annealing

Scheme de răcire:

$T(k)=T(0)/(k+1)$

$T(k)=T(0)/\ln(k+c)$

$T(k)=aT(k-1)$ $(a<1$, ex: $a=0.995)$

Obs. 1. T(0) se alege astfel încât la primele iterații să fie acceptate aproape toate configurațiile generate (pentru a asigura o bună explorare a spațiului soluțiilor)

2. Pe parcursul procesului iterativ este indicat să se rețină de fiecare dată cea mai bună valoare întâlnită

Simulated Annealing

Proprietăți de convergență:

Dacă sunt satisfăcute proprietățile:

- Pg(x(k+1)=x'|x(k)=x)>0 pentru orice x și x' (probabilitatea de trecere între oricare două configurații este nenulă)
- Pa(x(k+1)=x'|x(k)=x)=min{1,exp(-(f(x')-f(x))/T)} (probabilitate de acceptare de tip Metropolis)
- T(k)=C/lg(k+c) (schema logaritmică de răcire)

Atunci P(f(x(k))=f(x*)) -> 1 (x(k) tinde în probabilitate la minimul global x* când k tinde la infinit)

Simulated Annealing

Variante: alte probabilități de acceptare (Tsallis)

$$P_a(x') = \begin{cases} 1, & \Delta f \le 0 \\ (1-(1-q)\Delta f / T)^{1/(1-q)}, & \Delta f > 0, (1\text{-}q)\Delta f \le 1 \\ 0, & \Delta f > 0, (1\text{-}q)\Delta f > 1 \end{cases}$$

$$\Delta f = f(x') - f(x)$$

$$q \in (0,1)$$

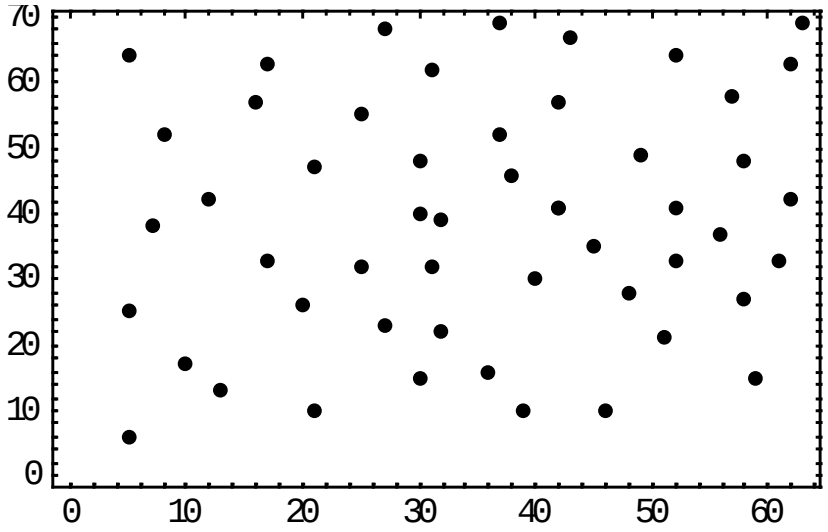
Simulated Annealing

Exemplu: problema comis voiajorului
(TSPLib: http://comopt.ifi.uni-heidelberg.de/software/TSPLIB95)
Instanta test: eil51 – 51 orase

Parametrii:

- 5000 iterații cu modificarea parametrului T la fiecare 100 de iterații
- T(k)=T(0)/(1+log(k))

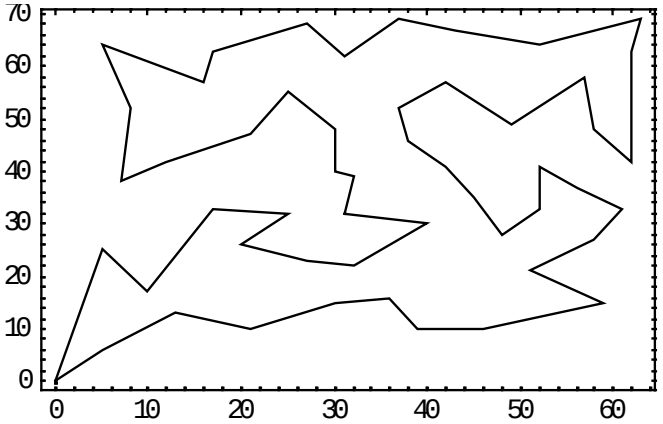
Distribuția oraselor



Simulated Annealing

Exemplu: problema comis voiajorului

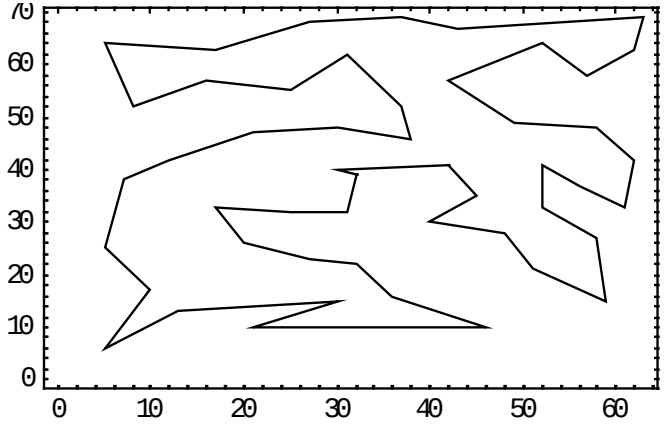
Instanta test: eil51 (TSPlib)



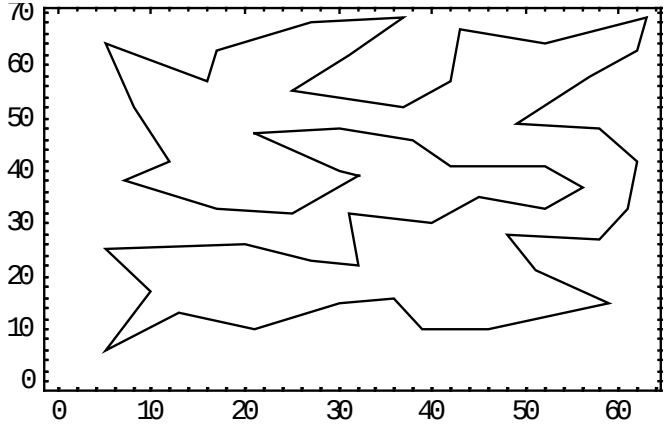
T(0)=10,cost=478.384

Cost minim: 426

T(0)=5, cost=474.178



T(0)=1, cost=481.32



Simulated Annealing

Exemplu: generarea orarelor

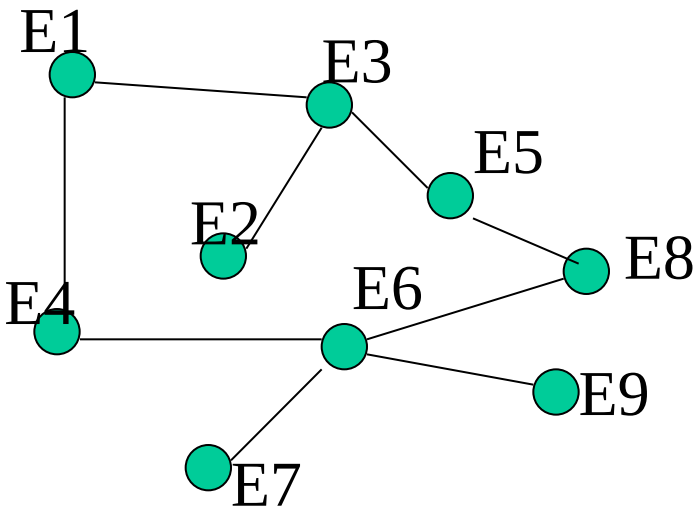
Problema: Se consideră un set de evenimente (ex: cursuri, examene), o mulțime de săli și un set de intervale de timp. Să se construiască un orar în care fiecare eveniment este asignat unei săli și unui interval de timp astfel încât să fie satisfacute o serie de restricții. Restricțiile pot fi:

- Puternice (obligatorii)
- Slabe (opționale)

	S1	S2	S3
T1	E1	E3	E9
T2	E5		E8
T3	E6	E4	
T4	E2		E7

Simulated Annealing

- Restricții puternice (solutia este acceptabilă doar dacă le satisface):
  - Un eveniment este planificat o singură dată
  - Intr-o sală este planificat un singur eveniment la un moment dat
  - Sala asignată corespunde caracteristicilor evenimentului
  - Nu sunt planificate simultan evenimente la care participă aceleași persoane



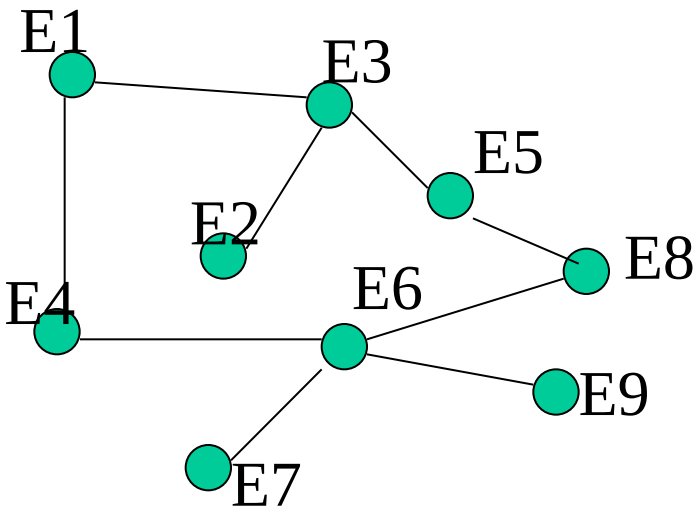
	S1	S2	S3
T1	E1	E3	E9
T2	E4		E8
T3	E6	E5	
T4	E2		E7

Graful conflictelor (două noduri conectate reprezintă evenimente ce nu pot fi planificate simultan)

Simulated Annealing

- Restricții slabe (soluția este mai bună dacă sunt satisfacute):
  - Nu sunt planificate mai mult de k evenimente succesive pentru același participant
  - Nu sunt participanți pt. care e planificat un singur eveniment/zi

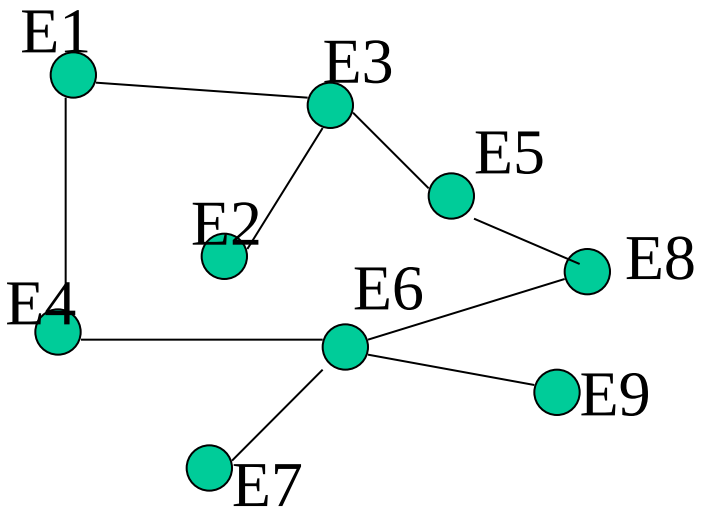
Idee: restricțiile slabe se transformă in criterii de optimizat (ex: numărul de participanti pentru care sunt planificate multe evenimente succesive și/sau un singur eveniment să fie cât mai mic)



	S1	S2	S3
T1	E1	E3	E9
T2	E4		E8
T3	E6	E5	
T4	E2		E7

Simulated Annealing

- Perturbarea unei configurații curente:
  - Transferul unui eveniment ce încalcă o restricție puternică într-o zonă liberă

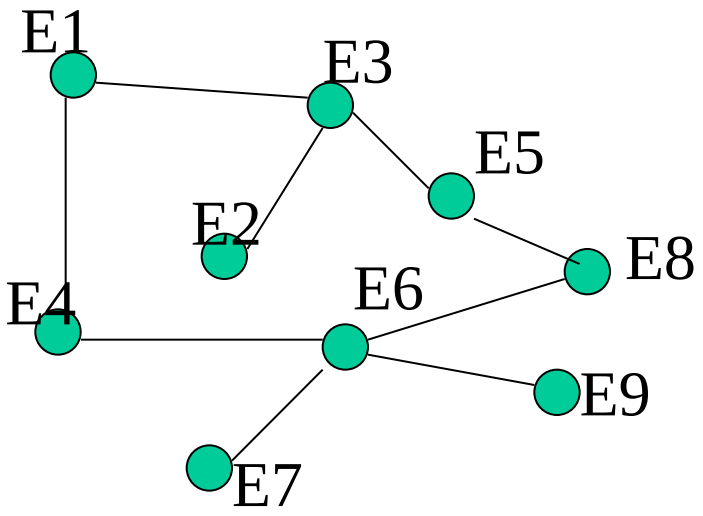


	S1	S2	S3
T1	E1	E3	E9
T2	E4		E8
T3	E6	E5	
T4	E2		E7

	S1	S2	S3
T1	E1		E9
T2	E4	E3	E8
T3	E6	E5	
T4	E2		E7

Simulated Annealing

- Perturbarea unei configurații curente:
  - Interschimbarea a două evenimente



	S1	S2	S3
T1	E1		E9
T2	E4	E3	E8
T3	E2	E5	
T4	E6		E7

	S1	S2	S3
T1	E1		E9
T2	E4	E3	E8
T3	E6	E5	
T4	E2		E7

Alte variante

Căutare bazată pe liste de configurații interzise (TS – Tabu Search)
Creator: Fred Glover (1986)

Scop: metoda de rezolvare a problemelor de optimizare combinatorială

Specific:

- Tehnică iterativă de căutare locală bazată pe explorarea vecinătății configurației curente (vecinătatea unei configurații se definește ca fiind mulțimea configurațiilor ce pot fi atinse din configurația curentă printr-o singură transformare; transformările posibile sunt specifice problemei)

- Utilizează o listă de configurații interzise care nu vor putea fi vizitate în următoarele iterații; lista tabu are o dimensiune limitată (e implementată ca listă circulară)

Alte variante

Căutare bazată pe liste de configurații interzise (TS – Tabu Search)

Structura generala:

Pas 1: se construiește o configurație inițiala

Pas 2: REPEAT

- Se selectează cel mai bun element din vecinătatea configurației curente care este acceptabil în raport cu lista tabu
- Dacă elementul selectat este suficient de bun atunci se adaugă la o arhiva cu elite
- Se actualizează lista tabu

UNTIL <conditie de oprire>

Obs:

1. Daca vecinătatea unei configurații este prea mare atunci nu se evaluează toate elementele ci doar o selecție a acestora
2. Pentru îmbunătățirea funcționarii se pot aplica periodic etape de intensificare și diversificare a căutarii

Alte variante

Căutare bazată pe liste de configurații interzise (TS – Tabu Search)

Intensificare:

Scop: exploatarea regiunilor ce par promițătoare

Mod de implementare:

- Se contorizează pentru fiecare componentă a configurației curente numărul de iterații consecutive în care a rămas nemodificată
- Se restartează procesul de căutare pornind de la cea mai bună configurație întâlnită considerând fixate componentele cu comportare bună (pentru care contorul are valori mari)

Alte variante

Căutare bazată pe liste de configurații interzise (TS – Tabu Search)

Diversificare:

Scop: explorarea regiunilor ce nu au fost vizitate

Mod de implementare:

- Se contorizează frecvența de utilizare, pe parcursul întregului proces iterativ, a valorilor corespunzătoare diferitelor componente
- Se restartează procesul de căutare de la configurații în care sunt plasate componente cu frecvența mică de utilizare; sau se penalizează scorul unei configurații folosind frecvențele asociate componentelor; relaxarea restricțiilor prin modificarea sistematică (creștere urmată de descreștere și invers) a ponderilor utilizate în funcția obiectiv care include restricțiile (construită prin tehnica penalizării)

Alte variante

Căutare bazată pe vecinătăți variabile (VNS - Variable Neighborhood Search) [Mladenovic, P. Hansen, 1997]

- Idee: utilizeaza o structură de vecinătăți V_1, V_2,...,V_{kmax} care care este explorată incremental; in cadrul fiecarei vecinatati căutarea se realizează utilizând o metodă de căutare locală
- Obs: Structura de vecinătăți ale unei configurații (soluții candidat) x se stabilesc în funcție de specificul problemei dar astfel încât dacă k1<k2 atunci elementele lui $V_{k1}(x)$ sunt mai „apropiate" de x decât elementele lui $V_{k2}(x)$
- Exemplu: pentru problema comis voiajorului $V_k(x)$ poate conține configurațiile obținute din x aplicând k interschimbări de elemente selectate aleator

Alte variante

Căutare bazată pe vecinătăți variabile (VNS - Variable Neighborhood Search) [Mladenovic, P. Hansen, 1997]

Structura generală

Inițializează x (aleator în spațiul de căutare)

k:=1

WHILE k<=kmax DO

selecteaza x' aleator din $V_k(x)$;

construieste x'' din x' aplicand o metoda de cautare locala

IF f(x'')<f(x') THEN x:=x''; k:=1

ELSE k:=k+1

Alte variante

Metode de căutare locală (directă, fără utilizarea derivatelor)

- Nelder – Mead
- Hooke - Jeeves