Laborator 5

Alocarea resurselor unui sistem (controlul sistemului de trafic feroviar)

1. Obiectivele laboratorului

- Însușirea unei metode de determinare a alocării resurselor unui sistem, folosind algoritmi genetici. Studiu de caz: generarea orarului trenurilor (*train schedule*),
- Însușirea unei metode de generare a rutelor trenurilor în sistemul feroviar, folosind algoritmi genetici. Generalizare pentru rute într-un graf oarecare.
- Însușirea modului de lucru pentru probleme cu constrângeri.

2. Studiu de caz I: determinarea alocării resurselor unui sistem folosind algoritmi genetici (generarea orarului trenurilor)

Alocarea de resurse într-un sistem este o problemă cu constrângeri multiple, referitoare la interdependențele care există între resurse și componentele care le utilizează, precum și între acestea și aspectul temporal. Deși există algoritmi de alocare a resurselor, găsirea soluției optime este dificilă și nu întotdeauna garantată.

În alte cazuri, calculul mai multor valori (care trebuie să lucreze împreună pentru îndeplinirea unui scop comun și care sunt dependente una de alta) este o problemă foarte specifică. De multe ori, singura soluție este un algoritm „euristic” ghidat, probabil, de cineva cu experiență în domeniu.

Algoritmii genetici pot fi de folos în aceste cazuri datorită componentei aleatoare. În continuare ne propunem să formulăm și să rezolvăm o problemă din această categorie.

2.1. Formularea problemei

Se dă sistemul de trafic feroviar din figura de mai jos. Există peroane (P_1, …, P_8), linii (L_1, …, L_4) și macazele situate la intersecțiile acestora (*interlocking*-urile I_1, …, I_6). Pe fiecare dintre aceste componente este permis să se afle maxim un tren la un moment dat. Evident, trenurile nu au voie să se ciocnească sub nici o formă. Există 3 rute predefinite pe care circulă trenuri (indicate în figură cu culori – galben: P_1, L_2, L_4, P_6, albastru: P_2, L_1, L_4, P_7, roșu: P_6, L_3, L_1, P_4).

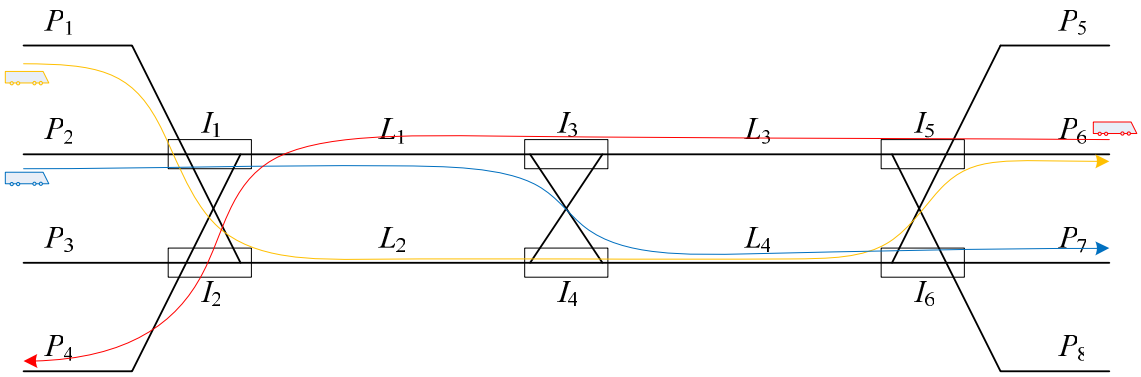

Figura 14. Sistemul de trafic feroviar

Timpii de parcurs pentru fiecare segment sunt indicați mai jos:

- Peroane (P_1, …, P_8) – 4 unități de timp
- Linii (L_1, …, L_4) – 6 unități de timp
- Interlocking-uri (I_1, …, I_6) – 1 unitate de timp

Se pune problema de a determina timpii de așteptare pentru fiecare tren, pe peroane și pe linii (exclus interlocking-uri), astfel încât fiecare tren să aștepte cât mai puțin și să nu existe ciocniri. Toate trenurile pornesc la momentul $t = 0$, dar pot exista întârzieri chiar la început.

2.2. Rezolvarea problemei

Soluția problemei va fi reprezentată de un vector cu 12 componente, reprezentând timpii de așteptare (*delay*) pentru fiecare peron sau linie din cadrul rutei celor 3 trenuri. Structura cromozomului este deci:

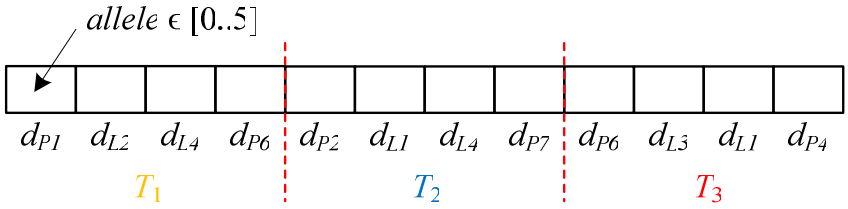

Figura 15. Structura cromozomului

Pentru această problemă, vom limita timpul de așteptare în intervalul [0, 5] unități de timp (genotipul) și vom realiza simularea sistemului cu un pas $\Delta t = 1$ unitate de timp. Alegerea întârzierilor nu garantează că trenurile nu se ciocnesc, de aceea trebuie luat și acest fapt în calculul performanței.

În caz că nu există ciocniri, performanța unui cromozom este calculată ca fiind timpul total de așteptare (d_{total}). Dacă există ciocniri, cromozomul este total neperformant și i se atribuie o valoare maximă:

if (*există_ciocniri*) **then** $J = MaxValue$ **else** $J = d_{total}$

2.3. Testare și concluzii

Găsirea soluțiilor valide și convergența

Dacă testăm algoritmul genetic cu funcția de performanță dată, vom observa că de multe ori nu generează nici o soluție fără ciocniri, adică cu performanța $J \neq MaxValue$. Aceasta se datorează faptului că toți cromozomii care generează ciocniri vor fi „la fel de neperformanți": există practic un salt al performanței care introduce o discontinuitate puternică a funcției de performanță în spațiul de căutare. Acest fapt poate conduce la imposibilitatea algoritmului de a ieși din zona de neperformanță. Pentru a evita acest lucru există mai multe variante:

- introducem suficientă varietate în populație de la început. În funcție de raportul dintre numărul de soluții cu ciocniri față de cele fără ciocniri, ar putea fi nevoie ca populația inițială să se apropie ca mărime de întreg spațiul de căutare, deci algoritmul să devină *brute force*.
- introducem o altă funcție de performanță care include, evident, timpul total de așteptare (d_{total}), dar și o componentă care penalizează puternic orice ciocnire. În formula de mai jos, $N_{ciocniri}$ este numărul de ciocniri în cadrul simulării, iar $f_{penalizare}$ este factorul scalar de penalizare.

$$J = d_{total} + f_{penalizare} \cdot N_{\text{ciocniri}}$$

Valoarea acestui factor va fi stabilită așa încât diferența procentuală între un cromozom care conține ciocniri și unul care nu conține ciocniri să fie de minim 80% (pentru a oferi operatorului de selecție probabilitate suficient de mare de a selecta cromozomul mai bun – o diferență de sub 50% nu ar garanta alegerea cromozomului mai performant). Cu ajutorul unui factor $f_{penalizare}$ suficient de mare, introducem practic intervale diferite de performanță (*clase de performanță*), care vizează comportamente diferite ale sistemului.

Timpul de așteptare d_{total} este în intervalul [0, 60], și avem nevoie de o diferență procentuală 80%. Rezultă deci $f_{penalizare} = 60 / (100\% - 80\%) = 300$.

Astfel, un cromozom cu 2 ciocniri va fi mai puțin performant decât unul cu o singură ciocnire (raport aproximativ de 50%), iar unul fără ciocniri va fi cu mult mai performant decât unul cu ciocniri (raport de 20%).

Lipsa soluțiilor valide

Dacă nu există o planificare a trenurilor conform condițiilor problemei, toți cromozomii vor avea cel puțin o ciocnire, deci $J > f_{penalizare}$. Din punctul de vedere al algoritmului, cea mai bună soluție va fi aceea cu J minim, deci cu cele mai puține ciocniri. Chiar dacă nu generează o soluție validă, algoritmul este totuși util pentru a oferi informații despre problemă: de exemplu, care tren este cel care generează ciocniri, sau care resursă este cea mai solicitată.

Oprirea algoritmului

Dacă, rulând algoritmul, descoperim că J devine mai mic decât $f_{penalizare}$ înseamnă că nu mai există ciocniri, astfel încât putem opri algoritmul după ce nu mai apar imbunatatiri semnificative. De obicei, se stabilește un prag de îmbunătățiri sub 5% – 10% pe parcursul a 10 generații consecutive.

3. Studiu de caz II: generarea rutelor trenurilor pentru sistemul feroviar

3.1. Formularea problemei

Considerând exemplul anterior, cu cele 3 rute prestabilite și timpii de așteptare pentru fiecare tren ca în figura de mai jos, se cere să se introducă un nou tren în plus (*charter*), care pornește din P_3 și ajunge pe oricare dintre peroanele din gara din dreapta. Evident, constrângerea împotriva ciocnirilor trebuie să se respecte cu strictețe și în acest caz. Așadar, trebuie calculată ruta și timpii de asteptare pe fiecare segment din rută pentru noul tren.

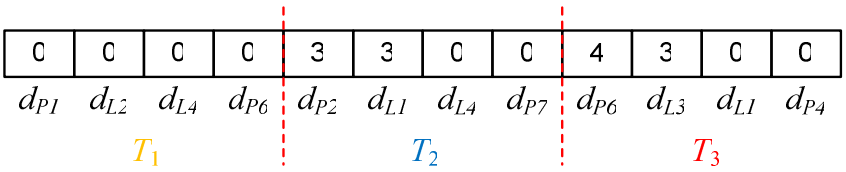

Figura 16. Întârzierile trenurilor planificate pentru fiecare segment

3.2. Rezolvarea problemei

Din cauza faptului că toți cromozomii trebuie să aibă lungime fixă, nu putem codifica ruta trenului în cromozom concatenând resursele care trebuie vizitate. Putem în schimb recurge la una dintre următoarele variante:

- gene care codifică poziția macazului din fiecare interlocking (în funcție de tipul macazului, gena va avea valori din mulțimea {0, 1} sau {0, 1, 2}) În acest caz, unele gene (interlocking-uri pe unde trenul nu circulă) nu vor avea efect asupra performanței. Este posibil ca unele soluții generate să nu fie valide, aspect analizat în problema precedentă.
- evaluarea dinainte a tuturor rutelor posibile, iar cromozomul va codifica index-ul rutei.
- observarea unor cazuri particulare ale problemei. De exemplu, putem observa că rutele posibile ale trenului sunt următoarele, unde perechile indicate în paranteze reprezintă variante posibile ale rutei:

  $P_3 \rightarrow (L_1, L_2) \rightarrow (L_3, L_4) \rightarrow (P_5, P_6, P_7, P_8)$

Observație: toate variantele prezentate mai sus pentru codificarea rutei în cromozom pot fi cu ușurință generalizate pentru un graf oarecare.

În cazul nostru, cromozomul va conține deci 3 gene care compun ruta, și încă 3 gene care vor reprezenta întârzierile pe fiecare dintre segmentele componente ale rutei (evident, nu este necesară o întârziere pe peronul de final). În figura de mai jos am indicat structura cromozomului și genotipul pentru fiecare genă în parte.

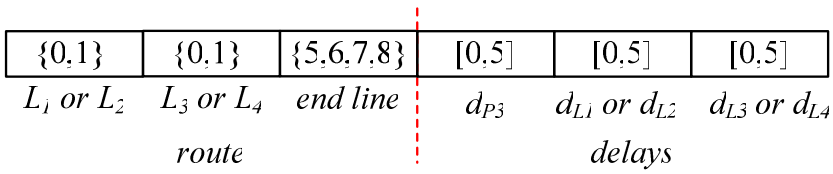

Figura 17. Structura cromozomului

Se observă că nu toate genele au același genotip, însă aceasta nu reprezintă un impediment pentru nici una din etapele buclei de evoluție a algoritmului genetic. Revăzând descrierea operatorilor de încrucișare (*crossover*) și mutație, observăm că:

- pentru încrucișare: fiecare genă își păstrează locul de ordine în cromozom: se schimbă câte o genă cu gena de pe poziția corespunzătoare a altui cromozom.
- pentru mutație: o genă ia o nouă valoare, aleasă din mulțimea genotipului acelei gene.

Funcția de performanță pentru această problemă are aceeași formă ca și în problema precedentă, penalizând puternic orice ciocnire a trenurilor și slab întârzierile.

4. <u>Exerciții:</u>

1. Implementați programul aferent studiului de caz I (determinarea întârzierilor). Testați programul mai întâi cu prima funcție de performanță, care nu penalizează diferențiat numărul de ciocniri. La a câta generație apare prima soluție fără ciocniri ?
2. Modificați funcția de performanță și folosiți $f_{penalizare} = 300$. Observați la ce generație scade brusc indexul *J*, semn că au fost eliminate ciocnirile care sunt penalizate puternic. Cum poate fi îmbunătățit algoritmul pentru a ajunge mai repede la acest rezultat ? Încercați diferite valori pentru $f_{penalizare}$ sau pentru funcția de performanță.
3. Implementați programul aferent studiului de caz II (determinarea rutei și a întârzierilor pentru un tren charter). Alegeți varianta indicată, observând cazul particular al problemei.
4. Testați programul de la exercițiul anterior inclusiv cu alte scenarii decât cel dat în Figura 16. Ce se întâmplă dacă scenariul folosit nu permite planificarea unui nou tren cu constrângerile date ?
5. Se dă un labirint sub forma unei matrice $N \times N$, cu valori 0 (pătrat liber) sau 1 (zid), si un prizonier pe unul din pătratele goale. Calculați cea mai scurtă rută de ieșire din labirint, folosind algoritmi genetici. Folosiți prima variantă indicată în capitolul 3.2.