Documentatie Proiect

Constantinescu Ana-Gabriela, 331

Descriere Lucrare Aleasa

Titlul lucrarii: Genetic Algorithm for Solving the Traveling Salesman Problem Using Neighbor-based Constructive Crossover Operator

Autori: Akshay Vyas si Dashmeet Kaur Chawla

Data publicarii: Aprilie 2018

Lucrarea aleasa propune un nou algoritm de incrucisare, numit Neighbor-based Constructive Crossover Operator, folosit pentru rezolvarea problemei comis voiajorului intr-un algortim genetic. Pentru determinarea performantei algoritmului, acesta va fi comparat cu alte 2 metode de incrucisare:

- SPCX – incrucisarea printr-un singur punct de taietura
- SCX – Sequential Constructive Crossover Operator

Algortimul genetic propus – descriere generala

```
GA() {
        initializare populatie;
        evaluarea populatiei;
        generatie = 0;
        cat timp conditia finala nu e indeplinita {
                generatie = generatie + 1;
                selectie parinti (elitism) cu Pr (probabilitate de reproducere);
                incrucisare cu Pc (probabilitate de incrucisare);
                mutatie cu Pm (probabilitate de mutatie);
                selectie supravietuitori;
                evaluare populatie;
        }
}
```

Selectie parinti

initializare lista parinti;

sortare populatie in functie de fitness (cel mai bun parinte se afla pe pozitia 0);

cat timp mai trebuie pusi parinti {

    r = rand();

    daca r < Pr atunci cel mai bun individ se adauga in lista parintilor;

    altfel, un individ random se adauga in lista parintilor;

}

Incrucisare parinti

SPCX – incrucisarea printr-un singur punct de taietura

r = un index random (de la 0 la lungimea solutiei - 1);

i = 0;

cat timp i < r {

    c1 si c2 primeste valoarea din p1, respectiv p2; }

cat timp i < lungimea solutiei {

    c1 si c2 primeste valoarea din p2, respectiv p1, daca valoarea este valida pentru copilul respectiv; }

i = 0;

cat timp i < r {

    c1 si c2 primeste valoarea din p2, respectiv p1, daca valoarea este valida pentru copilul respectiv; }

SCX – Sequential Constructive Crossover Operator

1. Se incepe de la nodul de pe pozitia 1 (c – nod curent).
2. Pe rand, se parcurg ambii parinti si se gaseste primul nod legitim (un nod care nu se gaseste inca in solutia copilului), pornind de la c. Daca niciun nod legitim nu se gaseste in vreunul dintre parinti, se parcurge lista {2, 3, …, n} si se considera primul nod legitim din lista.
3. Tinand cont ca in parintele 1 s-a gasit nodul x, iar in parintele 2 s-a gasit nodul y, se verifica distantele de la nodul c la nodurile x, respectiv y.
4. Daca distanta de la nodul c la x este mai mica, atunci nodul x se adauga in solutia copilului. Altfel, se adauga nodul y. Se repeta algortimul incepand cu pasul 2 pana cand solutia copilului rezultat este completa.

NCX – Neigbor-based Constructive Crossover Operator

1. Se incepe de la nodul de pe pozitia 1 (c – nod curent).
2. Se considera vecinii legitimi, din stanga si din dreapta, ai nodului c in fiecare din cei doi parinti. Daca niciunul dintre parinti nu contine un nod legitim, se considera primul nod legitim din multimea {2, 3, …, n}.
3. Tinand cont ca in parintele 1 s-au gasit nodurile x_1 si x_2, iar in parintele 2 s-au gasit nodurile y_1 si y_2, se verifica distantele de la nodul c la cele 4 noduri.
4. Nodul corespunzator costului minim se adauga in solutia copilului, acesta devenind noul nod c. Se reia algoritmul, pornind cu pasul 2, pana cand solutia copilului va fi completa.

Mutatie

Lucrarea mentioneaza folosirea mutatiei de tip 2-swap. In acest algoritm, se selecteaza 2 pozitii la intamplare si se interschimba orasele de pe acele 2 pozitii.

Selectia supravietuitorilor

Lucrarea nu mentioneaza cum se face selectia supravietuitorilor. Am decis sa folosesc selectia (μ+ λ), intrucat este selctia pe care am folosit-o eu intr-o tema de laborator si mi se pare ca a dat rezultate bune. In acest algoritm, se iau in considerare parintii si toti copiii acestora (inclusiv cei obtinuti prin mutatie) si se selecteaza cei cu fitnessul cel mai bun pentru a merge mai departe.

Valorile parametrilor folositi

- marimea populatiei: 50
- Pr: 10%
- Pc: 80%
- Pm: 10%
- numarul total de generatii: 1000

Excesul unei solutii

Pentru fiecare din datele de test folosite, valorile optime sunt cunoscute. Pentru fiecare best si average obtinute, s-a calculat si excesul acestor fitnessuri, prin urmatoarea formula:

$$exces = \frac{fitness\ obtinut - fitness\ optim}{fitness\ optim}$$

Astfel, idealul este ca valoarea excesului sa fie 0.

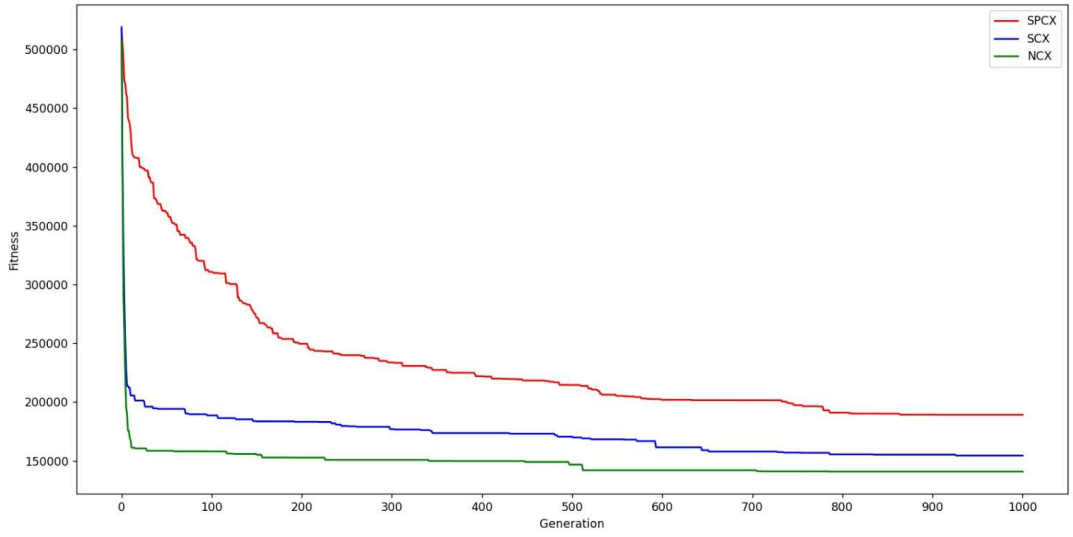
Rezultate replicate

TSP simetric – distantele de la orasul x la y, si invers, sunt intotdeauna egale.

Fisierele folosite pentru aceste dete sunt: eil51, pr76 si eil101. Rezultatele de pe fundalul galben sunt cele obtinute de mine, iar cele de pe fundal alb sunt luate din lucrare.

		NCX			SCX			SPCX		
n	Opt val	Best (e)	Avg (e)	Time	Best (e)	Avg(e)	Time	Best (e)	Avg(e)	Time
51	426	460(7.98)	473(11.03)	1.4	476(11.73)	501.9(17.81)	1.28	797(87.08)	932.5(118.89)	0.37
51	426	465(9.15)	495.3(16.28)	3.25	465(9.15)	493.9(15.93)	2.40	518(21.59)	566.9(33.08)	1.99
76	108159	123390 (14.08)	130222.9 (20.39)	2.39	137127 (26.78)	144131.8 (33.25)	2.02	262244 (142.46)	339919.9 (214.27)	0.54
76	108159	116168 (7.4)	133699.8 (23.61)	6.39	124487 (15.09)	136099.3 (25.83)	4.82	149227 (37.97)	174284.9 (61.14)	3.32
101	629	705(12.08)	735.2(16.88)	3.79	837(33.07)	852.8(35.58)	3.09	1860(195.7)	2101.6(234.11)	0.79
101	629	722(14.79)	784.3(24.69)	9.56	769(22.26)	812.4(29.16)	7.86	901(43.24)	1001.7(59.25)	4.63

Evolutia besturilor pentru cele 3 tipuri de incrucisari (pr76):

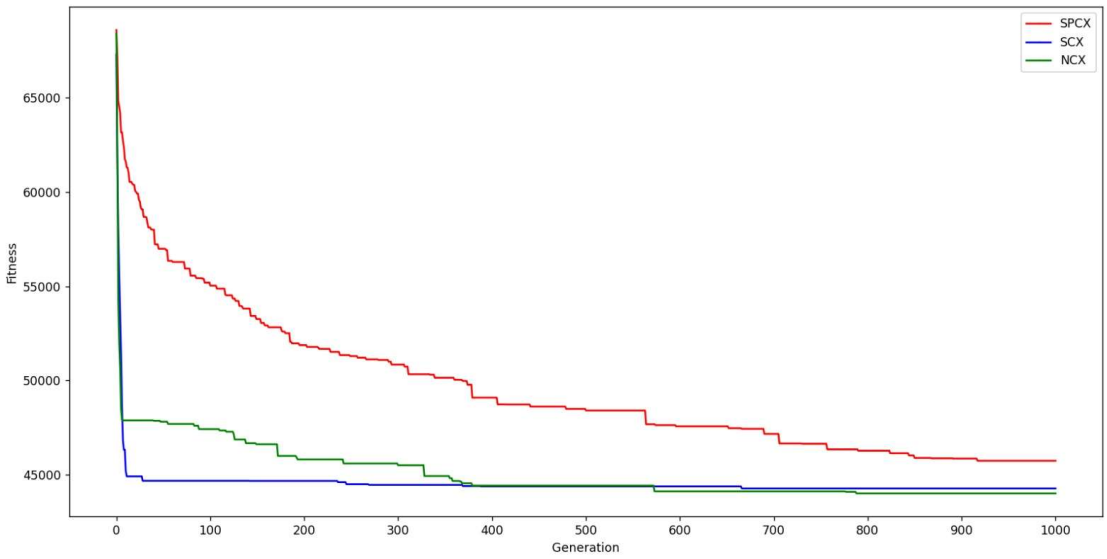


TSP asimetric – distanta de la orasul x la y este, in general, diferita de distanta de la y al x.

Fisierele folosite pentru aceste dete sunt: ftv33, ft70 si ftv 170. Rezultatele de pe fundalul galben sunt cele obtinute de mine, iar cele de pe fundal alb sunt luate din lucrare.

		NCX			SCX			SPCX		
n	Opt val	Best (e)	Avg (e)	Time	Best (e)	Avg(e)	Time	Best (e)	Avg(e)	Time
34	1286	1350(4.97)	1487.1(15.63)	0.79	1726(17.17)	1765.2(19.83)	0.65	2233(73.63)	2561(99.14)	0.27
34	1286	1460(13.53)	1568.3(21.95)	2.11	1418(10.26)	1538.3(19.61)	1.65	1620(25.97)	1790.7(39.25)	1.31
70	38673	41342(6.9)	42078.8(8.8)	2.22	42243(9.23)	43521.9 (12.53)	1.89	53980(39.58)	56904(47.14)	0.5
70	38673	42341(9.48)	44511.7(15.1)	5.45	42262(9.28)	43154.8 (11.59)	4	44299(14.55)	45805.3(18.44)	2.94
171	2755	3530(28.13)	3884.1(40.98)	8.34	4581(66.27)	4811(74.64)	6.88	16530(500)	17372.1(530.56)	1.79
171	2755	4575(66.06)	5418(96.66)	50.37	4500(63.34)	4994.5(81.29)	32.52	6786(146.31)	7579.6(175.12)	10.26

Evolutia besturilor pentru cele 3 tipuri de incrucisari (ft70):



Imbunatatire algoritm

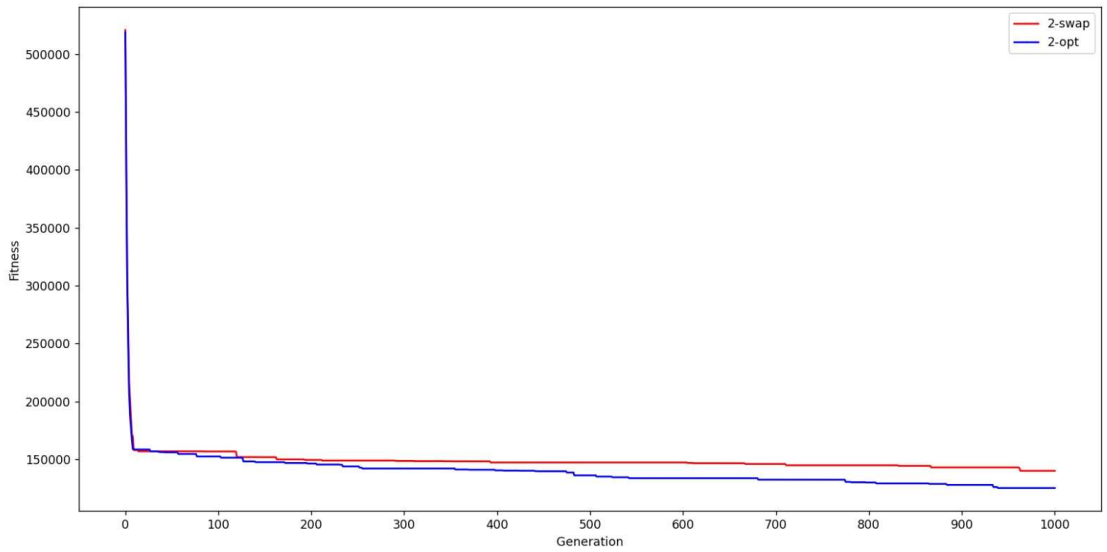
In loc de mutatia 2-swap, am ales sa folosesc mutatia 2-opt. Aceasta din urma, in loc sa schimbe doar locatia a 2 orase, inverseaza complet drumul dintre 2 orase. Am ales sa folosesc aceasta mutatie, intrucat eu am obtinut rezultate mai bune la temele de laborator folosind 2-opt. Am aplicat schimbarea cu mutatia 2-opt doar asupra algoritmului genetic care foloseste incurcisarea NCX.

TSP simetric

Rezultatele din coloana 2-swap sunt cele obtinute de mine, nu cele din lucrare.

		2-SWAP			2-OPT		
n	Opt_val	Best (e)	Avg (e)	Time	Best (e)	Avg(e)	Time
51	426	465(9.15)	495.3(16.28)	3.25	440(3.28)	453(6.36)	3.24
76	108159	116168 (7.4)	133699.8 (23.61)	6.39	109477(1.22)	115358(6.66)	6.23
101	629	722(14.79)	784.3(24.69)	9.56	656(4.29)	686.2(9.09)	9.34

Evolutia besturilor pentru cele 2 tipuri de mutatii (pr76):

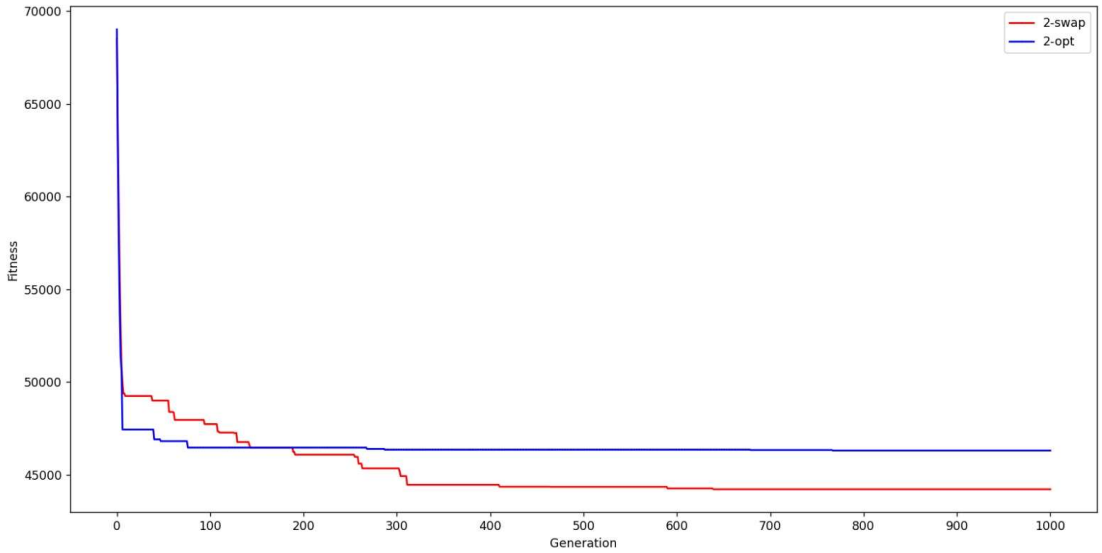


TSP asimetric

Rezultatele din coloana 2-swap sunt cele obtinute de mine, nu cele din lucrare.

		2-SWAP			2-OPT		
n	Opt_val	Best (e)	Avg (e)	Time	Best (e)	Avg(e)	Time
34	1286	1460(13.53)	1568.3(21.95)	2.11	1502(16.8)	1573.4(22.35)	1.99
70	38673	42341(9.48)	44511.7(15.1)	5.45	43499(12.48)	45284.4(17.1)	6.7
171	2755	4575(66.06)	5418(96.66)	50.37	5394(95.79)	6068.8(120.28)	47.61

Evolutia besturilor pentru cele 2 tipuri de mutatii (ft70):



Observatii

Compararea rezultatelor din lucrare cu cele obtinute de mine:

- TSP simetric si TSP asimetric: am obtinut rezultate mai bune pentru SPCX si SCX, dar un pic mai proaste pentru NCX

Compararea celor 3 tipuri de incrucisari (folosind datele exclusiv obtinute de mine):

- TSP simetric: rezultate NCX > rezultate SCX > rezultate SPCX (rezultatul asteptat – la fel au obtinut si autorii lucrarii)
- TSP asimetric: rezultate SCX > rezultate NCX > rezultate SPCX

Compararea celor 2 tipuri de mutatii (folosind doar incrucisarea NCX):

- TSP simetric: rezultate 2-opt > rezultate 2-swap lucrare > rezultate 2-swap proprii
- TSP asimetric: rezultate 2-swap lucrare > rezultate 2-swap proprii > rezultate 2-opt

Concluzii

Consider ca tipul TSP-ului (simetric sau asimetric) este foarte important atunci cand se scrie un algoritm genetic.

In cazul TSP-ului simetric, NCX este un tip de incrucisare care da rezultate mai bune decat SCX si SPCX, atat in lucrarea pe care mi-am bazat proiectul, cat si in rularile mele. In cazul TSP-ului asimetric, eu am obtinut rezultate mai bune pentru SCX, spre deosebire de lucrare, unde s-au obtinut rezultate mai bune tot pentru NCX. Nu pot spune exact de unde apare aceasta diferenta, dar printre posibilitati se numara:

- intelegerea gresita a algoritmului NCX

- diferente in construirea algoritmului (selectia supravietuitorilor nu a fost specificata in lucrare; de asmenea, nici selectia parintilor nu a fost bine detaliata)

Din punct de vedere al schimbarii aduse, desi mutatia 2-opt s-a dovedit a fi cea mai buna in cazul TSP-ului simetric, aceasta nu are sens in contextual unui TSP asimetric. La o mutatie 2-opt, inversarea unui intreg drum nu mentine distantele dintre noduri, astfel incat poate strica solutia intr-o mare masura.

Bibliografie

(1) Genetic Algorithm for Solving the Traveling Salesman Problem Using Neighbor-based Constructive Crossover Operator, Akshay Vyas si Dashmeet Kaur Chawla, Aprilie 2018
(2) Genetic Algorithm for the Traveling Salesman Problem using Sequential Constructive Crossover Operator, Zakir H. Ahmed