**Problema rucsacului**

**(Algoritm evolutiv)**

**Prezentare algoritm**

Algoritmul evolutiv pentru problema rucsacului a fost implementat pe baza unei reprezentări binare, folosind principiile OOP.

**A computer screen shot of a program code

Description automatically generated**

**population –** contine o lista de tupluri de lungime numar indivizi, unde fiecare tuplu este format dintr-o solutie, sub forma unei liste de orase, si fitness-ul soluției respective

**pasii se repeta de numar generatii ori:**

**parinti –** obtinuti prin selectie turnir (aleg random 2 indivizi din populatie

adaug in parinti individul cu fitness-ul mai bun(adica fitness ul mai mic)) repetat de numar indivizi ori

**p\_x –** obtinuta prin incrucisarea parintilor (order crossover)

* aleg 2 puncte de taietura
* aleg random 2 solutii din parinti
* primul descendent va fi la fel ca parinte 1 intre cele 2 puncte de taietura
* al doilea descendent va fi la fel ca parinte 2 intre cele 2 puncte de taietura
* in continuare se aplica algoritmul order crossover
* la final, descendentii sunt adaugati ca tuplu, alaturi de fitness-ul lor in p\_x

**p\_m –** lista care reprezinta descendentii dupa mutatie (mutatie tare), pentru fiecare cromozom al fiecarui individ, se genereaza un numar random[0,1], iar daca acel numar este mai mic decat probabilitatea de mutatie, cromozomul isi schimba valoarea (0->1 sau 1->0)

**populatia** generatiei urmatoare este formata din supravietuitorii generatiei precedente

* am selectat supravietuitorii folosind selectie proportionala cu fitness-ul, algoritmul ruletei.
* am concatenat intr-o lista populatia precedenta, descendentii si descendentii dupa mutatie, iar pe urma am eliminate duplicatele
* am calculat performanata totala a populatiei (suma fitnessurilor)
* am calculat probabilitatea de selectie pentru fiecare individ
* pentru fiecare individ am adaugat intr o lista probabilitatea sa de selectie si suma probabilitatilor tuturor indivizilor dinaintea lui
* pentru fiecare individ am generat un g aleator [0, 1] si pe baza algoritmului ruletei am adaugat in lista de supravietuitori, elementul corespunzator din lista de indivizi

**Algoritmul returneaza cel mai bun individ din ultima generatie .**

Pentru fiecare set de parametri, in fisierul de output citit de la tastatura va fi adaugat fitness-ul celui mai bun individ si fitness-ul mediu pentru fiecare generatie si la final cel mai bun individ din toate generatiile si calitatea medie per total. Pentru fiecare configuratie este reprezentat un grafic cu evolutia rezultatelor pentru fiecare generatie.

**Rezultate algoritm**

* **N** reprezintă numărul de indivizi din populatie
* **M** reprezintă numărul maxim de generații
* **P** reprezintă probabilitatea de mutație
* **Best** reprezintă valoarea celei mai bune soluții valide
* **Average** calitatea medie a soluțiilor valide generate
* **Timp executie** timpul necesar generării celor 10 soluții și determinării:

- celei mai bune soluții

- solutia medie

Următoarele date au fost obținute în urma rulării algoritmului pe setul de date din fișierul ***„rucsac-20.txt”.***

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| N | M | P | Best | Average | Timp executie(secunde) |
| 30 | 50 | 0.15 | 718 | 679 | 0.05040764808654785 |
| 30 | 50 | 0.5 | 699 | 665 | 0.04882621765136719 |
| 30 | 750 | 0.15 | 688 | 649 | 0.7200345993041992 |
| 30 | 750 | 0.5 | 711 | 664 | 0.7418534755706787 |
| 30 | 1000 | 0.15 | 711 | 674 | 0.9880490303039551 |
| 30 | 1000 | 0.5 | 716 | 666 | 1.1328859329223633 |
|  |  |  |  |  |  |
| 50 | 50 | 0.2 | 726 | 700 | 0.10674595832824707 |
| 50 | 50 | 0.65 | 726 | 707 | 0.10903716087341309 |
| 50 | 750 | 0.2 | 726 | 707 | 1.3399016857147217 |
| 50 | 750 | 0.65 | 726 | 699 | 1.4378485679626465 |
| 50 | 1000 | 0.2 | 726 | 709 | 1.7435863018035889 |
| 50 | 1000 | 0.65 | 726 | 696 | 1.9848594665527344 |
|  |  |  |  |  |  |
| 100 | 100 | 0.3 | 726 | 716 | 0.41977906227111816 |
| 100 | 100 | 0.85 | 726 | 719 | 0.45968031883239746 |
| 100 | 500 | 0.3 | 726 | 722 | 2.400163412094116 |
| 100 | 500 | 0.85 | 726 | 723 | 2.3499631881713867 |
| 100 | 1000 | 0.3 | 726 | 722 | 4.490735292434692 |
| 100 | 1000 | 0.85 | 726 | 723 | 4.520186424255371 |

Prin comparatie cu random search, random hill climbing sau tabu search ajungem la concluzia ca rezultatele obtinute in urma algoritmului evolutiv sunt mult mai calitative decat toate de pana acum.

In experimentele efectuate am rulat pentru acelasi numar de indivizi si acelasi numar de generatii cu probabilitate diferita. Concluzia este ca, in mare parte, la o probabilitate de mutatie mai mare, se obtin rezultate mai calitative.

De asemenea, schimband si numarul de indivizi, am ajuns la concluzia ca un numar mai mare de indivizi nu inseamna si obtinere de solutii mai calitative.

In comparatie cu mutatia slaba, mutatia tare genereaza solutii mai calitative.

Următoarele date au fost obținute în urma rulării algoritmului pe setul de date din fișierul ***„rucsac-200.txt”.***

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| N | M | P | Best | Average | Timp executie(secunde) |
| 30 | 50 | 0.15 | 132660 | 131663 | 0.43018221855163574 |
| 30 | 50 | 0.5 | 133087 | 131105 | 0.4039912223815918 |
| 30 | 750 | 0.15 | 132160 | 131171 | 6.4564597606658936 |
| 30 | 750 | 0.5 | 132825 | 130697 | 6.529995679855347 |
| 30 | 1000 | 0.15 | 131878 | 130903 | 7.600353717803955 |
| 30 | 1000 | 0.5 | 131847 | 130954 | 8.110183000564575 |
|  |  |  |  |  |  |
| 50 | 50 | 0.2 | 132693 | 131418 | 0.712090015411377 |
| 50 | 50 | 0.62 | 132541 | 131742 | 0.7170755863189697 |
| 50 | 750 | 0.2 | 131701 | 130987 | 10.325277090072632 |
| 50 | 750 | 0.62 | 132353 | 131784 | 10.59580397605896 |
| 50 | 1000 | 0.2 | 131841 | 131281 | 13.801595687866211 |
| 50 | 1000 | 0.62 | 131517 | 131003 | 13.69568157196045 |
|  |  |  |  |  |  |
| 100 | 100 | 0.3 | 132794 | 132167 | 3.1201252937316895 |
| 100 | 100 | 0.83 | 132901 | 131976 | 3.509394645690918 |
| 100 | 750 | 0.3 | 132764 | 132001 | 22.35070013999939 |
| 100 | 750 | 0.83 | 132163 | 131508 | 19.73425793647766 |
| 100 | 1000 | 0.3 | 132392 | 131572 | 25.669524908065796 |
| 100 | 1000 | 0.83 | 132913 | 131812 | 35.274736166000366 |

Concluziile sunt aceleasi ca si in cazul fisierului ***„rucsac-20.txt”.***