1. **Descriere algoritmi in pseudocod**

n-numarul de indivizi din populatie (setat in cod – linia 31)

m-numarul de generatii (citit de la tastatura)

pm-probabilitatea de mutatie (setat in cod – linia 43)

**Algoritm:**

Populatie = GenerateRandomPopulation ()

currentGeneration=0;

cat timp(currentGeneration< m)

Parinti= TournamentSelectionParents (Populatie, rand) // Populatie – lista care contine toata populatia, rand – obj de tip random.

Copii = CrossParents(Parinti, rand) // pentru generare copii se foloseste incrucisare printr-o singura taietura.

Mutatii = MutationOfCrossedParent(Copii, pm, rand) // pentru mutatii se foloseste strong mutation

Populatie = SelectSurvivors(Parinti, Copii, Mutatii) // pentru selectarea supravietuitorilor se inlocuiesc cele mai mici solutii din Parinti cu cele mai bune solutii din Copii si din Mutatii

currentGeneration ++

sf cat timp

1. **Tabel de comparative a celor 3 date**

Pentru Rucsac de 200

|  |  |  |  |
| --- | --- | --- | --- |
| **Nr** | Populatie=100, nr de generatii=5,  Pm=0.2 | Populatie=200, nr de generatii=5,Pm=0.5 | Populatie=400, nr de generatii=10, Pm=0.01 |
| 1. | 133016 | 132910 | 133343 |
| 2. | 133037 | 133642 | 133527 |
| 3. | 132936 | 132675 | 133217 |
| 4. | 132815 | 132602 | 133552 |
| 5. | 133122 | 132492 | 133512 |
| 6. | 133274 | 132955 | 133143 |
| 7. | 133398 | 132747 | 133805 |
| 8. | 133485 | 132549 | 133836 |
| 9. | 132903 | 132905 | 133294 |
| 10. | 132702 | 132487 | 133496 |
| **Best** | 133485 | 133642 | 133836 |
| **Avrage** | 133068 | 132796 | 133472 |

Cu toate ca am incercat weak mutation, strong mutation, diferite valori pentru populatie, generatii si p, valorile obtinute au fost tot aproapiate intre ele.

1. **Descriere**

Pentru selectia parintilor am folosit functia TournamentSelectionParents(Population, rand). Aceasta functie se foloseste de lista de Population pentru a cauta 5 elemente random, dintre care se va alege ca parinte cel cu fitnessul cel mai bun.

Pentru formarea liste de Copii am folosit functia CrossParents(Parinti, rand) care se foloseste de metoda incrucisarii printr-o singura taietura.

Pentru mutatii am folosit functia MutationOfCrossedParent(Copii, Pm, rand).

Aceasta functie se foloseste de algoritmul STRONG MUTATION pentru a forma mutatiile copiilor.

Toate solutiile obtinute prin incrucisare sau prin mutate sunt considerate corectate in cazul in care se depaseste greutate admisa, folosit functia ResolveMistake(elem, rand) care alege aleator pozitii in sir pentru a fi transformate in 0.

Pentru selectia supravietuitorilor am luat cele mai bune elemente din Copii si din Mutatii si le am suprascris peste cele mai rele elemente din Parinti.

(cele mai bune elemente = fitnessul cel mai mare)

(cele mai rele elemente = fitnessul ce mai mic)

Toate solutiile care depasesc greutatea admisa sunt corectate.