Studiul aprofundat al a modului de funcționare a algoritmului

După rularea algoritmului PSO, am constatat că nu am reușit să atingem soluția optimă căutată, deși în câteva rânduri ne-am apropiat de ea. Am identificat câteva posibile cauze ale acestui comportament, însă doar prin deducție logică. Vom urmări pas cu pas desfășurarea algoritmului în primele trei iterații. La fel ca data trecută, dacă veți replica exemplul, veți obține rezultate asemănătoare ca și comportament, dar diferite ca valori, din cauza componentelor aleatorii din calculul vitezei particulelor.

Inițializare

Deoarece valorile inițiale ale particulelor din matricea `pbest` (populația paralelă a optimului personal al fiecărei particule) și vectorul `gbest` (liderul roiului) nu vor fi folosite în niciun calcul, putem să le inițializăm cum dorim. Am inițializat `pbest` cu valori aleatorii, iar `gbest`, cu valori 0. Însă funcțiile de adaptare ale acestor particule trebuie să fie mari, pentru a fi înlocuite în prima iterație cu valori realiste, calculate. Am ales pentru valorile funcțiilor de adaptare `fpbest` și `fgbest` valoarea comună de 1000000. Am limitat numărul de zecimale afișate la 4, pentru a nu aglomera excesiv afișarea.

Deoarece populația inițială pentru teste are zece particule, același număr de particule se găsesc și în `pbest`.

	pbest								fpbest
Nr particula	litera 1	litera 2	litera 3	litera 4	litera 5	litera 6	litera 7	litera 8	funcția de adaptare
1	0.8147	0.1576	0.6557	0.7060	0.4387	0.2760	0.7513	0.8407	10000000
2	0.9058	0.9706	0.0357	0.0318	0.3816	0.6797	0.2551	0.2543	10000000
3	0.1270	0.9572	0.8491	0.2769	0.7655	0.6551	0.5060	0.8143	10000000
4	0.9134	0.4854	0.9340	0.0462	0.7952	0.1626	0.6991	0.2435	10000000
5	0.6324	0.8003	0.6787	0.0971	0.1869	0.1190	0.8909	0.9293	10000000
6	0.0975	0.1419	0.7577	0.8235	0.4898	0.4984	0.9593	0.3500	10000000
7	0.2785	0.4218	0.7431	0.6948	0.4456	0.9597	0.5472	0.1966	10000000
8	0.5469	0.9157	0.3922	0.3171	0.6463	0.3404	0.1386	0.2511	10000000
9	0.9575	0.7922	0.6555	0.9502	0.7094	0.5853	0.1493	0.6160	10000000
10	0.9649	0.9595	0.1712	0.0344	0.7547	0.2238	0.2575	0.4733	10000000
	gbest								fgbest
	0	0	0	0	0	0	0	0	10000000

Vitezele inițiale a particulelor se inițializează aleatoriu, cu valori foarte mici. Vitezele sunt diferite pentru fiecare particulă și pentru fiecare dimensiune paralelă de căutare (literă) dintr-o particulă.

vit

Nr particula	litera 1	litera 2	litera 3	litera 4	litera 5	litera 6	litera 7	litera 8
1	0.0035	0.0008	0.0016	0.0045	0.0011	0.0043	0.0085	0.0042
2	0.0083	0.0005	0.0079	0.0008	0.0096	0.0091	0.0062	0.0005
3	0.0059	0.0053	0.0031	0.0023	0.0000	0.0018	0.0035	0.0090
4	0.0055	0.0078	0.0053	0.0091	0.0077	0.0026	0.0051	0.0094
5	0.0092	0.0093	0.0017	0.0015	0.0082	0.0015	0.0040	0.0049
6	0.0029	0.0013	0.0060	0.0083	0.0087	0.0014	0.0008	0.0049
7	0.0076	0.0057	0.0026	0.0054	0.0008	0.0087	0.0024	0.0034
8	0.0075	0.0047	0.0065	0.0100	0.0040	0.0058	0.0012	0.0090
9	0.0038	0.0001	0.0069	0.0008	0.0026	0.0055	0.0018	0.0037
10	0.0057	0.0034	0.0075	0.0044	0.0080	0.0014	0.0024	0.0011

Populația inițială este populația test, cu zece indivizi, a fost generată cu două capitole în urmă. Ea are următoarea structură:

pop

particula	litera 1	litera 2	litera 3	litera 4	litera 5	litera 6	litera 7	litera 8
1	99	121	98	100	98	102	98	113
2	104	111	119	113	116	103	114	110
3	117	110	120	109	110	119	98	121
4	98	103	117	116	109	98	99	113
5	120	109	99	115	120	109	110	117
6	115	113	104	120	112	101	99	108
7	109	114	105	119	112	121	117	108
8	111	107	114	105	118	115	117	118
9	103	106	100	114	117	110	115	99
10	108	122	115	102	111	109	101	100

Se convertește cuvântul care trebuie ghicit („albastru”) în vectorul numeric al codurilor ASCII ale literelor care îl compun:

vref

litera 1	litera 2	litera 3	litera 4	litera 5	litera 6	litera 7	litera 8
97	108	98	97	115	116	114	117

Se inițializează factorul de inerție și se calculează rata de descreștere a acestuia de la o iterație la următoarea.

fi	dif
0.9	0.005

Iterația 1

Se recalculează factorul de inerție

fi=
0.8950

Se calculează funcțiile de adaptare ale particulelor din populația curentă.

	pop								fadapt
Nr particula	litera 1	litera 2	litera 3	litera 4	litera 5	litera 6	litera 7	litera 8	funcția de adaptare
1	99	121	98	100	98	102	98	113	69
2	104	111	119	113	116	103	114	110	68
3	117	110	120	109	110	119	98	121	84
4	98	103	117	116	109	98	99	113	87
5	120	109	99	115	120	109	110	117	59
6	115	113	104	120	112	101	99	108	94
7	109	114	105	119	112	121	117	108	67
8	111	107	114	105	118	115	117	118	47
9	103	106	100	114	117	110	115	99	54
10	108	122	115	102	111	109	101	100	88

Se recalculează optimul personal al fiecărei particule, optimul global al roiului (liderul) și funcțiile de adaptare ale acestora. În prima iterație, pozițiile `pbest` coincid cu prima poziție calculată a fiecărei particule, adică cu particulele din populația inițială, iar liderul `gbest` este particula cu cea mai bună funcție de adaptare din populația inițială. Soluția optimă căutată de algoritm trebuie să aibă funcția de adaptare 0, deoarece este identică cu cuvântul căutat. Așadar, vor fi considerate mai bune particulele cu funcții de adaptare mai mici. Funcțiile de adaptare ale particulelor inițiale din `pbest` aveau valori mult mai mari, deci populația `pbest` este înlocuită în întregime. Liderul de moment al roiului este particula numărul 8 din populația inițială.

	pbest								fpbest
Nr particula	litera 1	litera 2	litera 3	litera 4	litera 5	litera 6	litera 7	litera 8	funcția de adaptare
1	99	121	98	100	98	102	98	113	69
2	104	111	119	113	116	103	114	110	68
3	117	110	120	109	110	119	98	121	84
4	98	103	117	116	109	98	99	113	87
5	120	109	99	115	120	109	110	117	59
6	115	113	104	120	112	101	99	108	94
7	109	114	105	119	112	121	117	108	67
8	111	107	114	105	118	115	117	118	47
9	103	106	100	114	117	110	115	99	54
10	108	122	115	102	111	109	101	100	88
	gbest								fgbest
	111	107	114	105	118	115	117	118	47

Se recalculează vitezele și pozițiile particulelor. Pozițiile se validează prin limitare la valorile maximă sau minimă admise, pentru ca valorile elementelor să nu iasă din intervalul 97-122. În iterația 1, deoarece `pbest`-ul unei particule coincide cu particula, deplasarea se face doar în direcția `gbest`. Folosind Workspace, extragem în paralel, pe măsură ce sunt calculate, vitezele, particulele nevalidate și apoi particulele validate, rulând instrucțiunile pas cu pas. Obținem:

vit

Nr particula	litera 1	litera 2	litera 3	litera 4	litera 5	litera 6	litera 7	litera 8
1	13.8082	-1.6731	7.5144	3.5356	32.8487	0.4044	1.6425	1.6936
2	9.6223	-1.4676	-3.6778	-10.0092	3.1295	1.9551	5.5819	12.4118
3	-7.7266	-2.2670	-9.7362	-4.2606	5.6117	-7.5104	33.2889	-3.2929
4	5.8788	1.3725	-1.3613	-9.5773	5.6067	31.3972	15.4920	1.8565
5	-10.8432	-2.8367	6.6539	-2.3470	-1.1796	3.8266	5.9419	1.0200
6	-1.8958	-5.5051	19.2670	-16.3970	6.2612	6.4858	17.6010	12.4855
7	0.4014	-3.6613	6.0387	-19.0277	1.6394	-8.6472	0.0021	13.0781
8	0.0065	0.0040	0.0056	0.0086	0.0034	0.0050	0.0011	0.0077
9	15.6621	1.4255	14.0191	-8.4789	0.1215	6.8244	0.1713	2.7181
10	5.8427	-19.4669	-1.5943	2.7265	6.0603	9.9050	2.6731	4.7951

particulele nevalidate

Nr particula	litera 1	litera 2	litera 3	litera 4	litera 5	litera 6	litera 7	litera 8
1	113	119	106	104	131	102	100	115
2	114	110	115	103	119	105	120	122
3	109	108	110	105	116	111	131	118
4	104	104	116	106	115	129	114	115
5	109	106	106	113	119	113	116	118
6	113	107	123	104	118	107	117	120
7	109	110	111	100	114	112	117	121
8	111	107	114	105	118	115	117	118
9	119	107	114	106	117	117	115	102
10	114	103	113	105	117	119	104	105

particulele validate

Nr particula	litera 1	litera 2	litera 3	litera 4	litera 5	litera 6	litera 7	litera 8
1	113	119	106	104	122	102	100	115
2	114	110	115	103	119	105	120	122
3	109	108	110	105	116	111	122	118
4	104	104	116	106	115	122	114	115
5	109	106	106	113	119	113	116	118
6	113	107	122	104	118	107	117	120
7	109	110	111	100	114	112	117	121
8	111	107	114	105	118	115	117	118
9	119	107	114	106	117	117	115	102
10	114	103	113	105	117	119	104	105

Se observă cum particulele au fost „deplasate” către particula `gbest`. Depășirile intervalului admisibil s-au făcut doar „în sus”, dincolo de valoarea maximă admisibilă, și au fost corectate la validare.

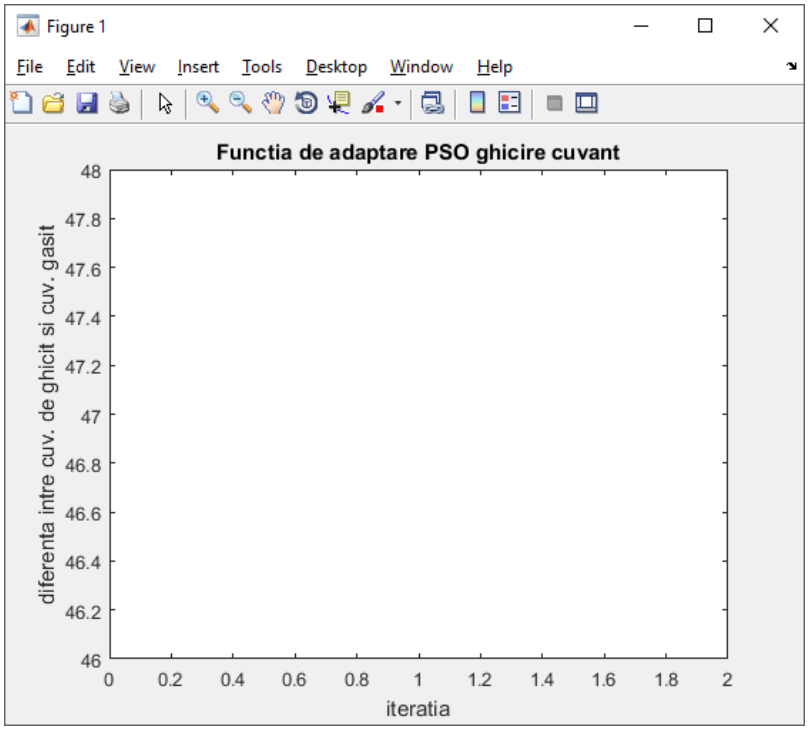
Afișări la final de iterație: liderul cu funcția sa de adaptare și cuvântul pe care îl codifică. Reprezentarea grafică conține doar un punct, care nu se vede pe un grafic de tip linie.

```
Iteratia:
     1

Solutia optima descoperita:
   111   107   114   105   118   115   117   118

Cuvant ghicit:
okrivsuv

Functia de adaptare a solutiei optime:
    47
```



Iterația 2

Se recalculează factorul de inerție.

Populația inițială a fost modificată substanțial. Este posibil ca printre noile particule să se găsească soluții mai bune, cu funcții de adaptare mai mici. Se recalculează funcțiile de adaptare, pentru a se evalua noile poziții ale particulelor din roi.

	pop								fadapt	fadapt
Nr particula	litera 1	litera 2	litera 3	litera 4	litera 5	litera 6	litera 7	litera 8	funcția de adaptare	în iterația anterioară
1	113	119	106	104	122	102	100	115	79	*69*
2	114	110	115	103	119	105	120	122	68	*68*
3	109	108	110	105	116	111	122	118	47	*84*
4	104	104	116	106	115	122	114	115	46	*87*
5	109	106	106	113	119	113	116	118	48	*59*
6	113	107	122	104	118	107	117	120	66	*94*
7	109	110	111	100	114	112	117	121	42	*67*
8	111	107	114	105	118	115	117	118	47	*47*
9	119	107	114	106	117	117	115	102	67	*54*
10	114	103	113	105	117	119	104	105	72	*88*

Se observă multe scăderi față de valorile funcțiilor de adaptare ale particulelor din iterația anterioară, însă există și cazuri în care noile soluții sunt la fel de bune precum cele anterioare (particula 2, 8) sau chiar mai proaste (particulele 1, 9). Însă avem un nou lider, cu valoarea funcției de adaptare 42 (particula 7), mai mică decât valoarea funcției de adaptare a liderului anterior, 47.

În pasul următor, la actualizarea populației paralele `pbest` vor fi înlocuite particulele 3, 4, 5, 6, 7, 10 și, de asemenea, va fi înlocuit și liderul `gbest`.

	pbest								fpbest
Nr particula	litera 1	litera 2	litera 3	litera 4	litera 5	litera 6	litera 7	litera 8	funcția de adaptare
1	99	121	98	100	98	102	98	113	69
2	104	111	119	113	116	103	114	110	68
3	109	108	110	105	116	111	122	118	47
4	104	104	116	106	115	122	114	115	46
5	109	106	106	113	119	113	116	118	48
6	113	107	122	104	118	107	117	120	66
7	109	110	111	100	114	112	117	121	42
8	111	107	114	105	118	115	117	118	47
9	103	106	100	114	117	110	115	99	54
10	114	103	113	105	117	119	104	105	72
	gbest								fgbest
	109	110	111	100	114	112	117	121	42

În pasul următor al iterației, deja unele particule din `pbest` nu mai coincid cu particulele din populație. Pentru aceste particule, căutarea nu se va mai face doar după `gbest`, ci și după `pbest`. De asemenea, s-a schimbat liderul. Obținem următoarele viteze, particule nevalidate și particule validate:

vit

Nr particula	litera 1	litera 2	litera 3	litera 4	litera 5	litera 6	litera 7	litera 8
1	1.5581	-11.1536	-4.0326	-6.8925	24.6068	20.0230	4.9642	1.0385
2	-2.0567	-0.8366	-3.5548	-7.2742	-7.2585	11.7237	2.2479	-8.6848
3	-6.4517	0.3519	-6.3660	-10.2493	3.9240	-5.5334	23.1890	3.1402
4	10.7745	3.8534	-4.9761	-14.9450	4.1499	20.2509	16.5609	4.7242
5	-8.9456	-0.2544	11.4330	-2.5216	-5.2257	2.5315	5.2250	1.9141
6	-2.1050	-2.5965	4.1199	-18.6811	1.8732	13.5182	14.4328	12.1754
7	0.3271	-2.9839	4.9215	-15.5076	1.3361	-7.0475	0.0017	10.6586
8	-1.0139	1.3475	-4.0024	-8.4370	-2.7529	-4.6791	0.0009	0.0466
9	-13.1109	5.0824	-17.1905	-7.2363	-0.9575	-7.6648	2.0318	3.3705
10	2.7875	-11.5485	-1.6399	-3.5809	0.7480	0.2717	13.2074	24.4583

particulele nevalidate

Nr particula	litera 1	litera 2	litera 3	litera 4	litera 5	litera 6	litera 7	litera 8
1	115	108	102	97	147	122	105	116
2	112	109	111	96	112	117	122	113
3	103	108	104	95	120	105	145	121
4	115	108	111	91	119	142	131	120
5	100	106	117	110	114	116	121	120
6	111	104	126	85	120	121	131	132
7	109	107	116	84	115	105	117	132
8	110	108	110	97	115	110	117	118
9	106	112	97	99	116	109	117	105
10	117	91	111	101	118	119	117	129

particulele validate

Nr particula	litera 1	litera 2	litera 3	litera 4	litera 5	litera 6	litera 7	litera 8
1	115	108	102	97	122	122	105	116
2	112	109	111	97	112	117	122	113
3	103	108	104	97	120	105	122	121
4	115	108	111	97	119	122	122	120
5	100	106	117	110	114	116	121	120
6	111	104	122	97	120	121	122	122
7	109	107	116	97	115	105	117	122
8	110	108	110	97	115	110	117	118
9	106	112	97	99	116	109	117	105
10	117	97	111	101	118	119	117	122

Vitezele particulelor se modifică substanțial. De asemenea, se înmulțesc valorile invalide din particule care trebuie corectate.

Deoarece corectarea se face prin limitarea la valoarea extremă admisă, vor apărea în populație multe valori de 97, corespunzătoare literei „a” și multe valori 122, corespunzătoare literei „z”. Aceste litere vor fi „favorizate”. Pentru ghicirea cuvântul „albastru”, e convenabil să favorizăm litera „a”, dar litera „z” e nedorită, deoarece nu există în cuvânt. Data trecută, am observat mulți de „a” și de „z” în soluțiile optime găsite, care nu erau soluția optimă globală. Aparent, cauza blocării algoritmului în aceste soluții pare să fie pare să fie modul în care se face validarea.

Un argument suplimentar în defavoarea acestui tip de validare este considerarea unui scenariu în care cuvântul ghicit, de exemplu „telefon”, nu conține nici litera „a”, nici litera „z”. În această situație, favorizarea acestor litere chiar îngreuiază căutarea, deoarece „a” și „z” ar trebui evitate.

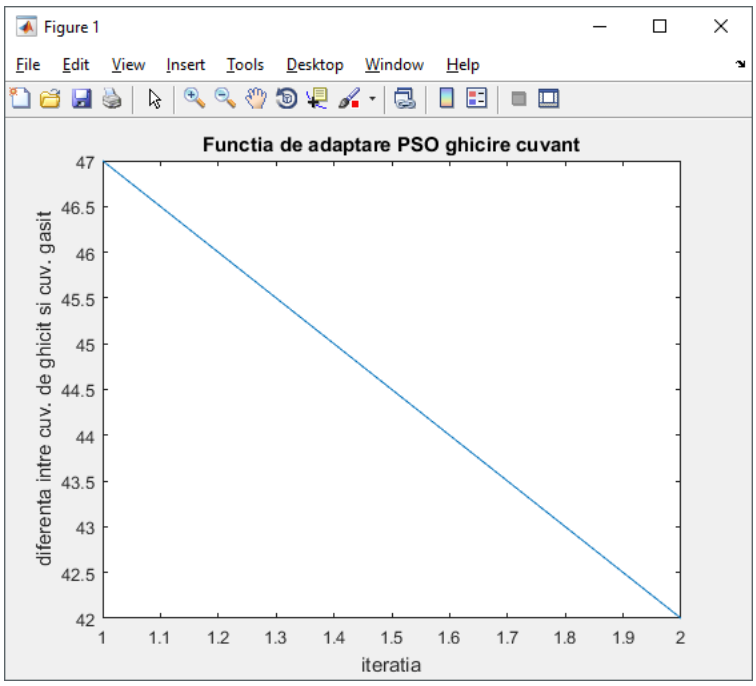
În afișările de la sfârșitul iterației 2, vedem cuvântul care reprezintă noul lider. Pe grafic, funcția de adaptare a liderului a scăzut de la 47 la 42.

```
Iteratia:
     2

Solutia optima descoperita:
   109   110   111   100   114   112   117   121

Cuvant ghicit:
mnodrpuy

Functia de adaptare a solutiei optime:
    42
```



Iterația 3

Se recalculează factorul de inerție

Se recalculează funcțiile de adaptare ale particulelor din populația actualizată:

	pop								fadapt	fadapt
Nr particula	litera 1	litera 2	litera 3	litera 4	litera 5	litera 6	litera 7	litera 8	funcția de adaptare	în iterația anterioară
1	115	108	102	97	122	122	105	116	45	*79*
2	112	109	111	97	112	117	122	113	45	*68*
3	103	108	104	97	120	105	122	121	40	*47*
4	115	108	111	97	119	122	122	120	52	*46*
5	100	106	117	110	114	116	121	120	48	*48*
6	111	104	122	97	120	121	122	122	65	*66*
7	109	107	116	97	115	105	117	122	50	*42*
8	110	108	110	97	115	110	117	118	35	*47*
9	106	112	97	99	116	109	117	105	39	*67*
10	117	97	111	101	118	119	117	122	62	*72*

Multe particule își îmbunătățesc poziția curentă. Dacă ea este mai bună decât cea din populația `pbest`, particula respectivă va fi copiată în `pbest`. De asemenea, avem un lider nou, particula cu numărul 8, care are funcția de adaptare 35.

Actualizarea pbest, gbest:

	pbest								fpbest	fpbest
Nr particula	litera 1	litera 2	litera 3	litera 4	litera 5	litera 6	litera 7	litera 8	funcția de adaptare	in iteratia anterioara
1	115	108	102	97	122	122	105	116	45	69
2	112	109	111	97	112	117	122	113	45	68
3	103	108	104	97	120	105	122	121	40	47
4	104	104	116	106	115	122	114	115	46	46
5	109	106	106	113	119	113	116	118	48	48
6	111	104	122	97	120	121	122	122	65	66
7	109	110	111	100	114	112	117	121	42	42
8	110	108	110	97	115	110	117	118	35	47
9	106	112	97	99	116	109	117	105	39	54
10	117	97	111	101	118	119	117	122	62	72
	gbest								fgbest	
	110	108	110	97	115	110	117	118	35	

Toate soluțiile noi descoperite de particulele care și-au îmbunătățit funcția de adaptare curentă sunt soluții optime personale noi și intră în `pbest`.

Vitezele, particulele noi nevalidate și particulele noi validate sunt:

vit

Nr particula	litera 1	litera 2	litera 3	litera 4	litera 5	litera 6	litera 7	litera 8
1	-4.8343	-8.8671	4.1337	-5.4796	8.7784	7.5130	19.8347	2.4903
2	-2.8971	-0.8993	-4.6879	-5.7466	-2.8574	0.3113	-3.6713	-0.3879
3	0.5995	0.2762	-0.6075	-8.0457	-3.1986	3.3761	8.8748	-3.3714
4	0.2449	1.8947	1.5988	-9.9684	-3.7480	12.1958	-6.7217	-2.2122
5	7.5357	2.6190	-2.6730	0.1911	3.4897	-5.2852	-5.4762	-0.8836
6	-2.3756	-0.2712	-15.7974	-14.3844	-1.8333	-4.3588	6.7268	2.7070
7	0.6500	-0.4653	-13.8393	-5.9241	-0.0067	10.1739	0.0013	7.1247
8	-0.7706	1.0241	-3.0419	-6.4121	-2.0922	-3.5561	0.0007	0.0354
9	-5.9139	-3.3696	1.9623	-8.8441	-2.2002	-4.6150	1.5340	19.8716
10	-10.9057	4.1006	-1.2639	-3.6525	-4.6152	-8.5136	9.9056	16.6684

particulele nevalidate

Nr particula	litera 1	litera 2	litera 3	litera 4	litera 5	litera 6	litera 7	litera 8
1	110	99	106	92	131	130	125	118
2	109	108	106	91	109	117	118	113
3	104	108	103	89	117	108	131	118
4	115	110	113	87	115	134	115	118
5	108	109	114	110	117	111	116	119
6	109	104	106	83	118	117	129	125
7	110	107	102	91	115	115	117	129
8	109	109	107	91	113	106	117	118
9	100	109	99	90	114	104	119	125
10	106	101	110	97	113	110	127	139

particulele validate

Nr particula	litera 1	litera 2	litera 3	litera 4	litera 5	litera 6	litera 7	litera 8
1	110	99	106	97	122	122	122	118
2	109	108	106	97	109	117	118	113
3	104	108	103	97	117	108	122	118
4	115	110	113	97	115	122	115	118
5	108	109	114	110	117	111	116	119
6	109	104	106	97	118	117	122	122
7	110	107	102	97	115	115	117	122
8	109	109	107	97	113	106	117	118
9	100	109	99	97	114	104	119	122
10	106	101	110	97	113	110	122	122

Din nou, sunt foarte multe valori care trebuie modificate la validare și multe litere „a” și „z” introduse în populație.

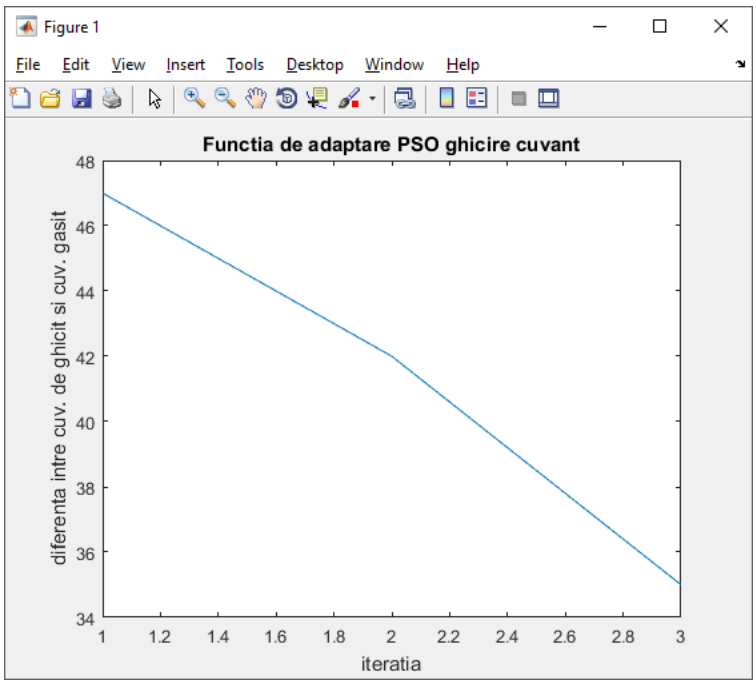
Afișări la sfârșit de iterație: noul lider și descreșterea funcției de adaptare a liderului.

```
Iteratia:
     3

Solutia optima descoperita:
   110   108   110    97   115   110   117   118

Cuvant ghicit:
nlnasnuv

Functia de adaptare a solutiei optime:
    35
```



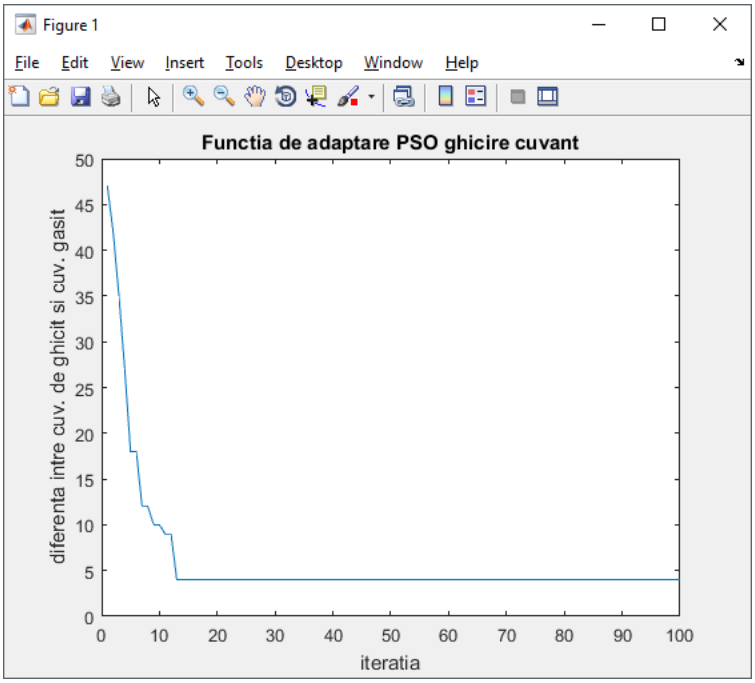
La sfârșitul celor 100 de iterații, obținem:

```
Iteratia:
   100

Solutia optima descoperita:
    97   108    97    97   115   118   115   117

Cuvant ghicit:
alaasvsu

Functia de adaptare a solutiei optime:
     4
```



Funcția de adaptare a liderului a continuat să scadă până la iterația 10 (aproximativ), după care a stagnat. Nu am ajuns la soluția optima căutată, valoarea funcției de adaptare a soluției găsite este 4, cu diferențe pe 3 litere din cuvânt.

Stagnarea timpurie a funcției de adaptare este frecventă la algoritmul PSO, mai ales când se folosesc populații mici. Mecanismul de modificare a soluțiilor face ca, treptat, particulele să înceapă să semene între ele. Dacă privim populația după cele 100 de iterații, ea arată așa:

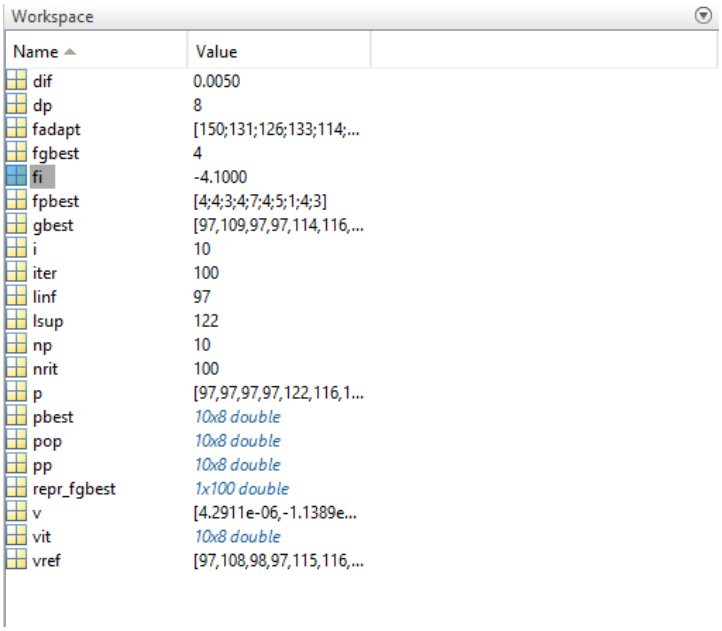
97	122	97	97	97	97	122	97
97	122	97	97	97	97	122	122
97	122	97	97	122	97	122	122
97	122	97	97	97	97	122	122
97	122	97	97	97	122	97	122
97	122	97	97	122	122	122	97
97	122	97	97	122	122	122	122
122	97	97	97	122	97	97	122
97	97	97	97	97	97	97	122
97	108	97	97	115	118	122	117

Cu excepția liderului, celelalte particule au doar valori 97 sau 122. Acesta este un semnal de alarmă. Ceva nu se petrece bine în algoritm. Probabil, validarea este responsabilă pentru această situație, dar ce anume forțează particulele dincolo de extreme? Deoarece particulele noi se obțin din viteze, să vedem ce valori au vitezele finale.

-7.7E+11	8.85E+10	-3.9E+07	-6331949	-1.1E+11	-3.74E+25	2.69E+25	-4.14E+16
-6.1E+08	91963143	-9.1E+07	-1.5E+07	-4.1E+12	-8.7E+08	1.36E+13	1.25E+13
-2E+09	76342602	-5.6E+07	-3.7E+07	4.20E+18	-1.1E+09	1.43E+09	3.55E+10
-1.2E+11	1.11E+09	-1.6E+08	-7.2E+07	-3.8E+11	-1E+09	3.44E+25	1.19E+11
-7.1E+11	1.35E+12	-1.4E+08	-1.4E+08	-9.2E+13	1.62E+10	-6.9E+12	8.86E+13
-8.6E+08	21770532	-1.4E+08	-1.7E+08	5.80E+25	5.81E+25	1.9E+13	-4.5E+12
-1.9E+09	4.16E+10	-1.6E+07	-5.9E+07	8.47E+25	6.93E+25	7.41E+25	8.29E+25
1.84E+08	-7.6E+07	-2.8E+08	-7.4E+07	8.34E+25	-1.07E+26	-4.5E+11	1.07E+26
-1.1E+07	-1.2E+08	-1.4E+08	-5.2E+07	-1.4E+10	-1.15E+26	-1E+11	6.20E+20
1.55E-05	0.001008	3.86E-06	1.36E-06	-0.00078	-0.00239	1.42E+26	-0.10911

Aceste valori ale vitezei nu sunt normale, sunt mult prea mari. Inerția ar fi trebuit să reducă viteza către finalul procesului iterativ, dar acest lucru nu se vede, valorile sunt enorme.

Analizând în Workspace rezultatele calculate după ultima iterație, se observă o valoare ciudată a factorului de inerție:



Valoarea finală a factorului de inerție este -4.1, deși ar fi trebuit să fie 0.4. Căutând în codul programului, găsim imediat sursa problemei.

Factorul de inerție este inițializat corect, înainte de procesul iterativ, dar este actualizat greșit. Actualizarea se face nu o data la fiecare iterație, ci de fiecare dată când se extrage câte o particulă din populație pentru recalcularea vitezei.

```
55
56      % vector functii de adaptare fgbest pentru reprezentare grafica
57      repr_fgbest=[];
58
59  -   for iter=1:nrit
60          % calcul functie de adaptare
61  -       [fadapt]=calcul_functie_adaptare(np,pop,vref);
62
63          % calcul pbest si gbest
64  -       [pbest,fpbest,gbest,fgbest]=actualizare_best(pop,fadapt,pbest,fpbest,gbest,fgbest);
65  -       pp=[];
66
67  -       for i=1:np % pentru fiecare individ din populatie
68              % recalculeaza factor inertie
69  -           fi=fi-dif;
70              % recalculeaza viteza particulei
71  -           v=fi*vit(i,:)+2*rand(1,dp).*(pbest(i,:)-pop(i,:))+2*rand(1,dp).*(gbest-pop(i,:));
```

Pentru o populație de 10 particule, actualizarea `fi` se face de zece ori într-o iterație. De aceea `fi` scade atât de mult. La final, are valoarea

0.9-(100(iterații)*10(indivizi)*0.005(dif))=0.9-5=4.1

corect ar fi

0.9-(100(iterații)*0.005(dif))=0.9-0.5=0.4

Dacă înmulțim viteza curentă cu o inerție cu valori mai mari decât 1 sau mai mici decât -1, riscăm să amplificăm mult valorile obținute pentru noua poziție a particulei, deoarece valorile vitezei se înmulțesc cu diferențe dintre particule, care pot lua valori x10. Drept consecință, când recalculăm poziția particulei, valorile elementelor vor rezulta dincolo de extreme, iar validarea le va limita la valorile extreme, ceea ce explică aspectul ciudat al populației din ultima iterație.

Pentru a corecta greșeala, mutăm actualizarea inerției imediat după începutul iterației:

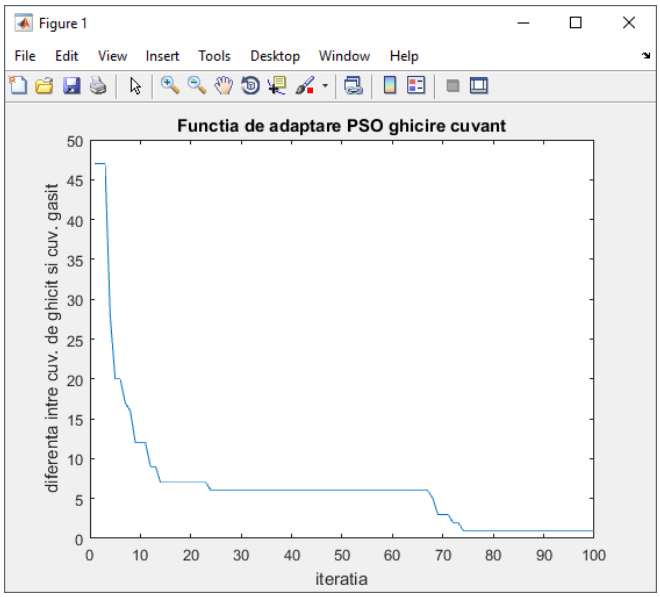
```
56      % vector functii de adaptare fgbest pentru reprezentare grafica
57      repr_fgbest=[];
58
59  -   for iter=1:nrit
60          % recalculeaza factor inertie
61  -       fi=fi-dif;
62
63          % calcul functie de adaptare
64  -       [fadapt]=calcul_functie_adaptare(np,pop,vref);
65
66          % calcul pbest si gbest
67  -       [pbest,fpbest,gbest,fgbest]=actualizare_best(pop,fadapt,pbest,fpbest,gbest,fgbest);
68  -       pp=[];
69
70  -       for i=1:np % pentru fiecare individ din populatie
71              % recalculeaza viteza particulei
72  -           v=fi*vit(i,:)+2*rand(1,dp).*(pbest(i,:)-pop(i,:))+2*rand(1,dp).*(gbest-pop(i,:));
73
```

Asemenea erori, deși pot părea minore, au uneori o influență majoră asupra rezultatelor. Ele pot apărea din neatenție sau din raționamente logice greșite și pot scăpa ușor neobservate, dacă nu se face testarea programului și analiza critică a rezultatelor obținute. În limba engleză, ele se numesc “bugs” (“gândaci”).

Să vedem dacă bug-ul descoperit influența negativ rezultatele algoritmului. Rulăm programul de zece ori, cu 100 de iterații.

Nr rulare	Cuvânt găsit	funcție de adaptare
1.	alaastzu	9
2.	albastru	0
3.	alaastrz	6
4.	alaastru	1
5.	alaastru	1
6.	alaastru	1
7.	alaastru	1
8.	alaaszru	7
9.	alaaszrz	12
10.	alaastru	1

Am reuşit să atingem o dată optimul global, şi de multe ori ne-am apropiat de el, însă există şi rulări relativ slabe. Un alt lucru pozitiv este că, de exemplu, la ultima rulare, graficul modificării funcției de adaptare arată mult mai bine. Nu se mai observă stagnarea anterioară.



Vitezele sunt și ele mici, conform așteptărilor:

-3.43E-19	2.01E-07	-3.81E-19	-2.40E-19	-6.88E-07	-1.80E-14	1.61E-10	1.46E-06
-4.33E-19	-0.03475	-1.83E-19	-1.43E-19	-8.26E-07	1.810739	-6.18E-10	-1.44149
-2.26E-19	-1.71821	-4.60E-19	-1.05E-19	-0.94523	0.716891	4.26E-13	4.12E-07
-1.05E-19	-5.37E-09	-5.14E-20	-3.30E-20	2.47E-13	1.11E-10	1.32E-10	0.009632
-1.52E-19	-1.31E-05	-8.40E-19	-5.00E-20	-8.09E-17	9.92E-19	1.47E-18	-1.83E-05
-2.87E-19	1.34E-06	-3.05E-19	-7.98E-20	3.82E-07	-9.36E-14	-4.65E-14	1.17E-08
-3.12E-19	-4.70E-08	-5.25E-19	-1.13E-20	1.03E-20	-6.78E-21	-8.34E-21	-0.88536
-6.93E-20	-0.0002	-4.52E-19	1.41E-20	-4.99E-10	-1.80E-11	-7.77E-06	-1.25225
-3.73E-19	0.004568	-4.07E-19	-5.77E-20	-1.22096	-1.97E-13	-0.02158	0.000359
-3.69E-19	-0.0004	-3.31E-19	-1.32E-19	-3.13E-07	2.800115	1.69E-14	-6.11E-07

Particulele din populație nu mai au valori extreme, dar aproape toate sunt identice:

97	108	97	97	115	116	114	117
97	108	97	97	115	116	114	117
97	107	97	97	113	117	114	117
97	108	97	97	115	116	114	117
97	108	97	97	115	116	114	117
97	108	97	97	115	116	114	117
97	108	97	97	115	116	114	115
97	108	97	97	115	116	114	117
97	108	97	97	115	116	114	117
97	108	97	97	115	117	114	117

Deoarece nu mai vedem valorile extreme atât de des, deocamdată apelăm la soluția cea mai comodă, de a crește numărul de indivizi din populație. Nu vom mai pleca de la populația test, ci vom regenera de fiecare dată populația inițială cu valori noi. Vom folosi o populație de 30 de indivizi.

```
% nr particule in populatie
np=30;
…..
% generare populatie
pop=round(linf+rand(np,dp)*(lsup-linf));
```

Rulăm algoritmul de zece ori, cu 100 de iterații, apoi încercăm şi cu mai mult, 500 de iteraţii Obținem:

Nr rulare	Cuvânt găsit, 100 it	funcție de adaptare, 100 it	Cuvânt găsit, 500 it	funcție de adaptare, 500 it
1.	alaastru	1	alaastrz	6
2.	alaastrz	6	alaastru	1
3.	albastru	0	alaastru	1
4.	alaastru	1	alaastru	1
5.	alaastru	1	albaszru	6
6.	alaastru	1	alaastru	1
7.	alaastru	1	alaastru	1
8.	alaastru	1	alaastru	1
9.	albastru	0	alaastrz	6
10.	alaastrz	6	alaastru	1

Rezultatele s-au îmbunătăţit, am găsit de două ori optimul global, dar toate soluţiile cu funcţia de adaptare 1 sau 6 „suferă” din cauza literelor „a” , respectiv „z”. Poate, schimbând metoda de validare, obţinem rezultate mai bune. 500 de iteraţii nu par să aducă beneficii, ba chiar dimpotrivă. Creşterea numărului de iteraţii reduce rata de descreştere pe iteraţie a inerţiei, ceea ce , se pare nu ajută algoritmului nostru.

Vom modifica metoda de validare a particulelor. În loc să limităm valoarea depășită la valoarea extremă admisă, vom regenera respectiva valoare aleatoriu în intervalul (97-122). Vom crea pentru aceasta o funcție `validare2` care arată astfel:

```
function [p]=validare2(dp,p,linf,lsup)
% Validarea unei particule cu valori iesite din limitele admisibile
% in algoritmul PSO ghicire cuvant:
% valorile depasite sunt inlocuite valori aleatorii din intervalul valid
%
% Date de intrare:
% dp - lungimea unei particule
% p - patricula supusa validarii, vector linie
% linf - limita inferioara admisibila pentru valorile elementelor
%        unei particule, numar
% lsup - limita inferioara admisibila pentru valorile elementelor
%        unei particule, numar
% Date de iesire:
% p -  particula dupa validare

 for j=1:dp % pentru toate elementele particulei modificate
    if (p(j)<linf) || (p(j)>lsup)
        p(j)=round(linf+rand*(lsup-linf));
    end;
end;
```

Schimbăm apelul validării cu apelul către această funcție, rulăm din nou algoritmul de zece ori, cu 100 de iterații și 30 de particule.

Nr rulare	Cuvânt găsit	funcție de adaptare
1.	bjdbrsrr	11
2.	bpcdsprt	14
3.	dkbdssru	8
4.	dkcaruqw	10
5.	flabuuru	10
6.	bliastqt	10
7.	bpcertrt	12
8.	bnbcsrss	10
9.	aiabstqq	10
10.	doadrtrv	12

Obținem cele mai proaste rezultate de până acum. Această metodă de validare nu ne avantajează.

Încercăm o a treia metodă de validare. În loc să regenerăm aleatoriu particulele cu valori depăşite, păstrăm valorile anterioare, cele din particula originală, peste care s-a aplicat viteza. Pentru aceasta, avem nevoie de particula respectivă ca valoare suplimentară de intrare în funcţie.

Funcţia arată astfel:

```
function [p]=validare3(dp,p,pinit,linf,lsup)
% Validarea unei particule cu valori iesite din limitele admisibile
% in algoritmul PSO ghicire cuvant:
% valorile depasite sunt inlocuite cu valorileddin particula initiala
%
% Date de intrare:
% dp - lungimea unei particule
% p - patricula supusa validarii, vector linie
% pinit - particula din care provine particula modificata
% linf - limita inferioara admisibila pentru valorile elementelor
%        unei particule, numar
% lsup - limita inferioara admisibila pentru valorile elementelor
%        unei particule, numar
```

```
% Date de iesire:
% p -  particula dupa validare


for j=1:dp % pentru toate elementele particulei modificate
    if (p(j)<linf) || (p(j)>lsup)
        p(j)=pinit(j);
    end;
end;
```

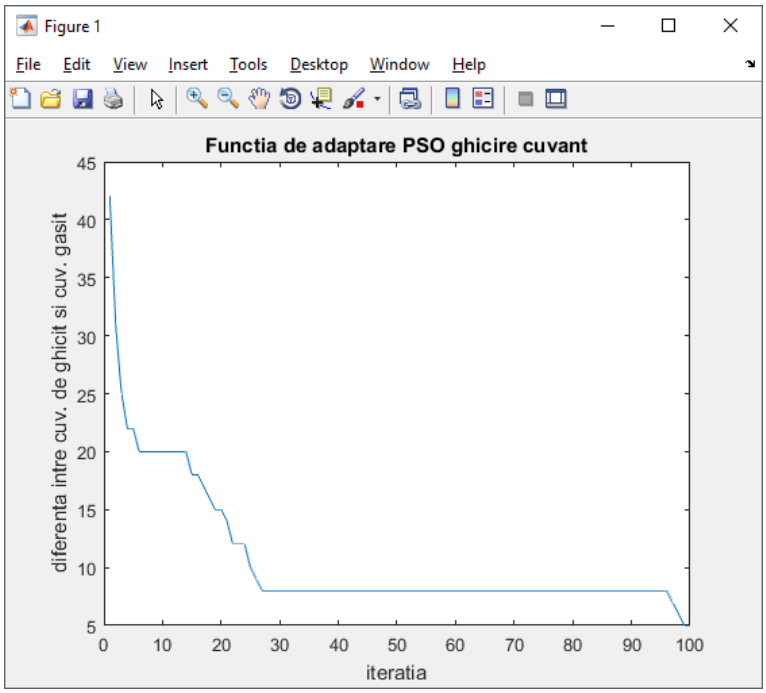
iar apelul ei în program se face astfel:

```
[p]=validare3(dp,p,pop(i,:),linf,lsup);
```

Rulăm din nou algoritmul de zece ori, cu 100 de iterații și 30 de particule.

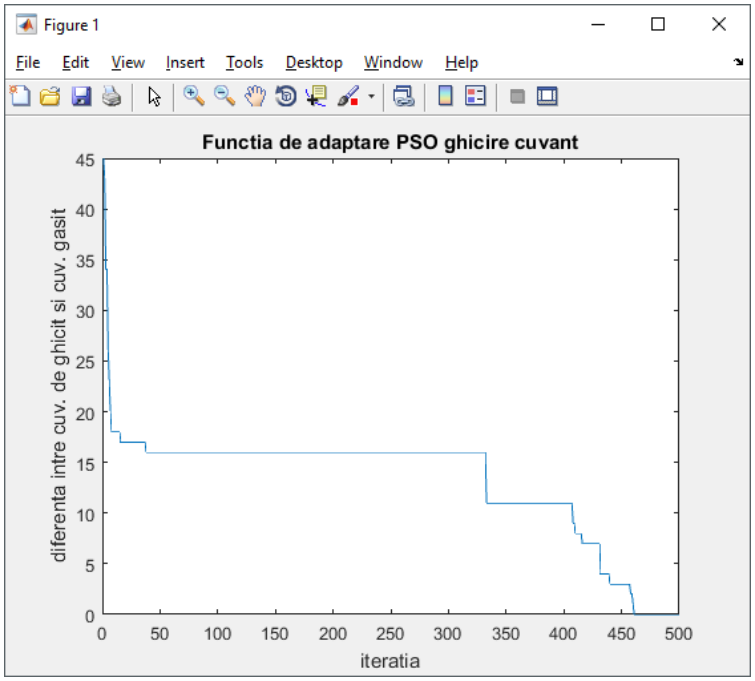
Nr rulare	Cuvânt găsit	funcție de adaptare
1.	aldattsu	4
2.	claastsu	4
3.	albastsu	1
4.	alaastru	1
5.	albastru	0
6.	albastru	0
7.	blcastqu	3
8.	alacsusu	5
9.	alaastsu	2
10.	albasrrt	3

Aparent, rezultatele par mai proaste decât cu prima formă de validare, dar se observă ceva pe grafic: pentru una dintre soluţiile cu funcţie de adaptare 5, ultima actualizare, una semnificativă, s-a făcut în ultimele iteraţii, iar acesta nu e singurul caz. Mai există loc pentru îmbunătăţiri ale soluţiei găsite. Încercăm cu 500 de iteraţii:



Nr rulare	Cuvânt găsit	funcție de adaptare
1.	albastru	0 (iteraţia 400)
2.	albastru	0 (iteraţia 370)
3.	albastru	0 (iteraţia 350)
4.	albastru	0 (iteraţia 470)
5.	albastru	0 (iteraţia 450)
6.	albastru	0 (iteraţia 470)
7.	albastru	0 (iteraţia 425)
8.	albastru	0 (iteraţia 460)
9.	albastru	0 (iteraţia 480)
10.	albaturu	2 (iteraţia 500)

Obţinem un scor aproape perfect, 9 din 10, Fapt interesant, algoritmul în general stagnează multe iteraţii la valori mari ale funcţiei de adaptare, dar, către sfârşit, converge rapid către soluţia optimă globală:

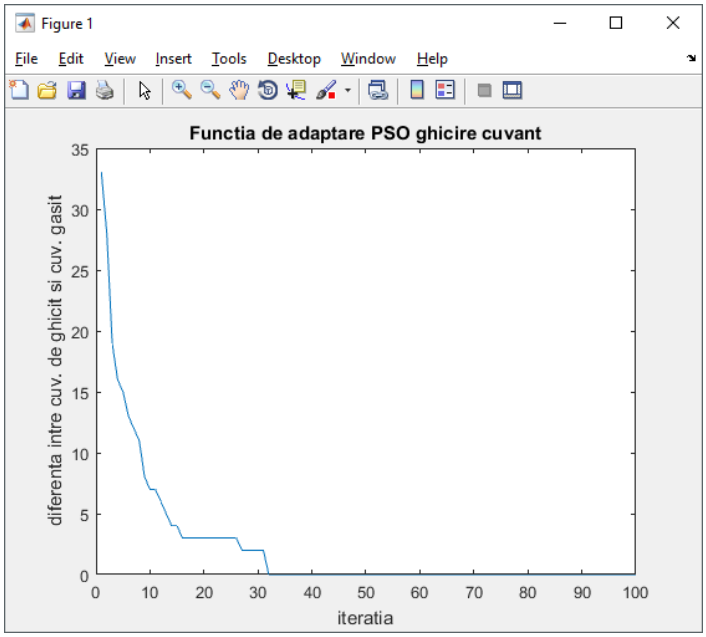


Printr-un accident, pe parcursul încercărilor, am descoperit un fapt curios. Viteza de convergenţă se măreşte la sub 100 de iteraţii dacă în algoritmul principal se renunţă la actualizarea vectorului vitezei după fiecare iteraţie. Viteza se recalculează cu formula cunoscută, dar, după calculul poziţiei particulei, nu se introduce modificată în matricea de viteze.

Dacă comentăm această linie:

```
% inlocuieste particula veche din populatie cu particula noua
% si viteza veche cu viteza noua
pop(i,:)=p;
vit(i,:)=v;
```

de foarte multe ori, graficul adaptării funcţiei arată aşa:



Se atinge optimul global după nici 40 de iteraţii, dintr-un total de 100 programate. Însă adaptarea vitezei după fiecare iteraţie face parte din algoritmul standard, renunţarea la ea ar fi neobişnuită. Această comportare ar trebui investigată pe mai multe tipuri de probleme, pentru a se vedea dacă e doar un caz particular sau o tendinţă generală.

Pentru a testa algoritmul în condiţii mai grele, îl vom pune să ghicească un cuvânt mai lung, “masacasasimultibani” (`dp`=19 litere).

Zece încercări, 1000 de iteraţii, 30 de particule, cu algoritmul care include recalcularea vitezei. Primele cinci încercări, le facem cu valorile standard pentru inerţie, între 0.9 - 0.4, iar ultimele 5, după analiza primelor cinci grafice obţinute, în care algoritmul scade puternic în zona de inerţie mică, de la sfârşitul iteraţiilor, le rulăm cu valoarea inerţiei între 0.9 - 0.2

Nr rulare	Cuvânt găsit	funcție de adaptare
1.	masaeasatimultibani	3
2.	masacasasimujtibani	2
3.	mbsacasasimultibani	1
4.	masacasatimultigani	6
5.	masacasasinultibani	1
6.	masacasasimultibani	0
7.	masacasasimultibani	0
8.	masacasasimultibani	0
9.	masacasasimultibani	0
10.	masacasasimultibani	0

Concluzie: cu inerţie mică, adică muncă mai multă, putem avea masă, casă şi mulţi bani !

Forma finală a algoritmului PSO este (pentru cuvântul „albastru”):

(am modificat puţin iniţializarea, pentru ca lungimea cuvântului de ghicit să fie calculată automat de program)

```
curatenie
clear;
clc;

% nr particule in populatie
np=30;

% numarul de iteratii
nrit=500;

% cuvantul ghicit:
cvg='albastru';

% dimensiunea cuvantului cautat (a unei particule)
dp=length(cvg);

% populatia initiala pbest
pbest=rand(np,dp);
% functiile de adaptare ale populatiei pbest
fpbest=10e6*ones(np,1);

% liderul initial al roiului
gbest=zeros(1,dp);
% functia de adaptare initiala a liderului
fgbest=10e6;

% vitezele initiale
vit=0.01*rand(np,dp);

% limite valori elemente individ
linf=97;
lsup=122;

% generare populatie
pop=round(linf+rand(np,dp)*(lsup-linf));

% vector numeric cuvant cautat (vector de referinta)
vref=double(cvg);

% factor initial de inertie
fi=0.9;
% rata de actualizare a factorului de inertie
dif=(0.9-0.4)/nrit;

% vector functii de adaptare fgbest pentru reprezentare grafica
repr_fgbest=[];

for iter=1:nrit
    % recalculeaza factor inertie
    fi=fi-dif;

    % calcul functie de adaptare
    [fadapt]=calcul_functie_adaptare(np,pop,vref);
```

```
    % calcul pbest si gbest
[pbest,fpbest,gbest,fgbest]=actualizare_best(pop,fadapt,pbest,fpbest,gbest,
fgbest);

    pp=[];
    for i=1:np % pentru fiecare individ din populatie
        % recalculeaza viteza particulei
        v=fi*vit(i,:)+2*rand(1,dp).*(pbest(i,:)-
pop(i,:))+2*rand(1,dp).*(gbest-pop(i,:));

        % recalculeaza pozitie particula si rotunjeste valorile
        p=round(pop(i,:)+v);

        % validare particula
        [p]=validare3(dp,p,pop(i,:),linf,lsup);

        % inlocuieste particula veche din populatie cu particula noua
        % si viteza veche cu viteza noua
        pop(i,:)=p;
        vit(i,:)=v;
    end;

    % afisari la sfarsitul fiecarei iteratii
    disp ('Iteratia:');
    disp(iter);
    disp('Solutia optima descoperita:');
    disp(gbest); % codurile ASCII
    disp('Cuvant ghicit:');
    disp(char(gbest)) % cuvantul
    disp('                    ')
    disp('Functia de adaptare a solutiei optime:');
    disp(fgbest);
    disp('                    ')

    % reprezentare grafica
    % adauga valoarea curenta a fgbest la vectorul pentru reprezentarea
    % grafica
    repr_fgbest=[repr_fgbest fgbest]; % adauga valoare in vector
    plot(repr_fgbest); % realizare grafic
    xlabel('iteratia'); % eticheta axa x
    ylabel('diferenta intre cuv. de ghicit si cuv. gasit'); % eticheta axa
y
    title ('Functia de adaptare PSO ghicire cuvant'); % titlu grafic
    drawnow;
    %pause; % asteapta apasarea unai taste pentru a continua
end;
```

Acesta apelează trei funcţii:

```
function [fadapt]=calcul_functie_adaptare(np,pop,vref)
% Calculul functiei de adaptare pentru PSO ghicire cuvant,
% diferenta absoluta dintre vectorul codurilor ASCII ale cuvantului
% gasit de algoritm si vectorul codurilor ASCII ale cuvantului de ghicit
%
% Date de intrare:
% np - numarul de particule din populatie
% pop - populatia curenta (matrice cu np linii)
% vref - vectorul numeric al codurilor ASCII ale cuvantului de ghicit
```

```
%
% Date de iesire:
% fadapt - functiile de adaptare ale particulelor din populatia curenta
%                  - vector coloana cu np elemente


% initializare vector functii de adaptare (coloana)
fadapt=zeros(np,1);

% calcul diferente
for i=1:np % pentru fiecare particula din populatie
    % extragere particula din populatie
    vact=pop(i,:);
    % calcul diferenta absoluta fata de vectorul de referinta
    df=sum(abs(vref-vact));
    % scriere diferenta in vectorul functiilor de adaptare
    fadapt(i)=df;
end;



function
[pbest,fpbest,gbest,fgbest]=actualizare_best(pop,fadapt,pbest,fpbest,gbest,
fgbest)
% Date de intrare:
% pop - populatia curenta, matrice
% fadapt - functiile de adaptare ale populatiei curente, vector coloana
% pbest - populatia paralela personal best pentru particule, matrice
% fpbest - functiile de adaptare ale populatiei pbest, vector coloana
% gbest - particula lider, vector linie
% fgbest - functia de adaptare a particulei lider, numar
%
% Date de iesire:
% matricele si vectorii pbest,fpbest,gbest,fgbest, dupa actualizare


% gaseste particulele care trebuie inlocuite
inloc=find(fadapt<fpbest);

% actualizeaza matricea pbest si vectorul fpbest
pbest(inloc,:)=pop(inloc,:);
fpbest(inloc,:)=fadapt(inloc,:);

% verifica daca s-a gast un noul lider gbest
lid=find(fpbest==min(fpbest));

% daca am gasit altul mai bun, inlocuiesc liderul
if fpbest(lid(1))<fgbest
    fgbest=fpbest(lid(1));
    gbest=pbest(lid(1),:);
end;
```

```
function [p]=validare3(dp,p,pinit,linf,lsup)
% Validarea unei particule cu valori iesite din limitele admisibile
% in algoritmul PSO ghicire cuvant:
% valorile depasite sunt inlocuite cu valorileddin particula initiala
%
% Date de intrare:
% dp - lungimea unei particule
% p - patricula supusa validarii, vector linie
% pinit - particula din care provine particula modificata
% linf - limita inferioara admisibila pentru valorile elementelor
%        unei particule, numar
% lsup - limita inferioara admisibila pentru valorile elementelor
%        unei particule, numar
% Date de iesire:
% p -  particula dupa validare


for j=1:dp % pentru toate elementele particulei modificate
    if (p(j)<linf) || (p(j)>lsup)
        p(j)=pinit(j);
    end;
end;
```

Observații finale

- Problema rezolvată de algoritm este conținută în calculul funcțiilor de adaptare și în validare.
- Actualizarea particulelor și a liderului se face după același mecanism, indiferent de problemă. Nu contează cum se obține valoarea funcției de adaptare, ci valoarea în sine, dacă este mai mică sau mai mare.
- Dacă se schimbă funcția de adaptare și cea de validare, algoritmul poate fi aplicat pentru orice problemă care modelează soluțiile sub formă de vectori.

Recapitulare

Am observat pas cu pas, vreme de trei iterații, mecanismul de funcționare al algoritmului PSO.

Am testat algoritmul cu diferite combinații de parametri, încercând să ajungem la soluția optimă. Pe parcursul testelor, am descoperit și am corectat un bug în program. Cu tipul de validare potrivit, mici reglaje de parametri şi un număr suficient de iteraţii, algoritmul descoperă aproape de fiecare dată soluţia căutată.

SFÂRSIT