A Modified Particle Swarm Optimization Algirithm

Vasilescu Vlad-Mihai
May 8, 2020

Abstract

In acest raport voi prezenta algoritmul genetic Modified Particle Swarm Optimization (MPSO) cat si efectul noului parametru denumit **inertie** asupra cazului concret de optimizare a unei functii obiectiv. Rezultatele experimentelor vor fi comparate cu cele obtinute de cei care au introdus pentru prima oara acest algoritm (1).

1 Introducere - Particle Swarm Optimization

Algoritmii genetici reprezinta o familie de algoritmi inspirati din teoria evolutiei, utilizand conceptul de evolutie a unei populatii (de solutii) pentru a oferii solutii la anumite probleme. La un anumit moment exista o populatie de solutii pentru problema de fata, fiecare individ din aceasta populatie temporara reprezentat un punct in spatiul de cautare al solutiilor determinat de problema curenta.
Aceasta colectie de indivizi denumita populatie reprezinta generatia curenta, adica generatia iteratiei de cautare curente, iar pentru a crea o noua generatie trebuie introduse concepte adecvate de recombinare a acestor indivizi. Evident, scopul urmarit este obtinerea de indivizi din ce in ce mai ”buni” de la iteratie la iteratie. Indivizii se compara intre ei prin intermediul unei functii de ”fitness”, care in cazul nostru va reprezenta functia obiectiv pentru care se doreste un estimat al optimului global.

Particle Swarm Optimization (PSO) reprezinta un exemplu din aceasta clasa de algoritmi, prin care indivizi evolueaza in timp tinand cont atat de pozitiile sale anterioare cele mai bune, cat si de pozitia globala cea mai buna din intreaga populatie de indivizi (”cooperare” intre indivizi).
Fiecare particula este reprezentata de un punct intr-un spatiu multidimensional, unde dimensionalitatea reprezinta defapt numarul de variabile necasare pentru a caracteriza functia obiectiv aferenta. Fie D dimensionalitatea functiei obiectiv, atunci pozitia fiecarei particule i poate fi reprezentata printr-un vector D-dimensional de forma:

$$X_i = (X_{i1}, ..., X_{iD}),$$

reprezentand solutia data de individul i la un anumit moment de timp. Reprezentand, prin aceeasi modalitate, vectorul pozitiei celei mai bune pentru individul i cu $\mathbf{P}_i$, vectorul pozitiei globale cele mai bune cu **G** (dintre toate particulele), si vectorul ”vitezelor” curente de deplasare pentru individul i si pentru fiecare dimensiune cu $\mathbf{V}_i$, rezulta ecuatiile de generare a noilor indivizi :

$$V_{ik} = V_{ik} + c1 \cdot random_1 \cdot (P_{ik} - X_{ik}) + c2 \cdot random_2 \cdot (G_k - X_{ik})$$

$$X_{ik} = X_{ik} + V_{ik}, \forall k = \overline{1, D}$$

unde random1 si random2 reprezinta 2 realizari particulare a unei variabile aleatoare repartizate uniform in intervalul [0, 1] (valorile nu sunt neaparat egale), iar c1 si c2 sunt 2 constante care, conform (2), trebuie alese aproape de valoarea intreaga 2, astfel incat proportiile pentru cea de-a 2-a si cea de-a 3-a parte a ecuatiei de update pentru viteza V_{ik} sa fie aproximativ 1 (cum media statistica a lui c1 si c2 este aprox. 0.5, o inmultire cu 2 va duce la o dublare a mediei statistice).

Astfel, se observa cum pentru calculul vitezei se tine cont atat de pozitia cea mai buna a particulei curente (P_{ik}) cat si de pozitia globala cea mai buna dintre toti indivizii, ditre toate iteratiile anterioare (G_k). Ultima ecuatie reprezinta ecuatia de deplasare a solutiei data de individul i, prin intermediul vitezei de deplasare calculate anterior (pozitia pe fiecare dimensiune este modificata conform vitezei pe care o are particula la acea iteratie pe acea dimensiune).

2 Modified Particle Swarm Optimization

Dupa cum se observa, fiecare individ evolueaza conform "propriei experiente" (pozitia lui cea mai buna), cat si conform informatiei globale cu privire la cea mai buna pozitie existenta dintre toate particulele. Astfel, cautarea unei stari optime (relativ la problema de fata) se compune in acelasi timp dintr-o cautare locala si una globala.

Pentru anumite tipuri de probleme este de dorit o cautare locala mai pronuntata, adica pozitia fiecarei particule este influentata mai mult de cea mai buna pozitie anterioara a particulei de fata, in timp ce pentru alte probleme este de dorit o influenta mai mare a informatiei globale asupra miscarii particulelor. In acest sens a fost introdus un nou parametru denumit **inertie** ce are rolul de a controla trade-offul dintre cautarea locala si cea globala. Acest parametru este identificat prin variabila **w** si intervine in calculul noii viteze de deplasare a particulei astfel:

$$V_{ik} = w \cdot V_{ik} + c1 \cdot random_1 \cdot (P_{ik} - X_{ik}) + c2 \cdot random_2 \cdot (G_k - X_{ik})$$

Acest parametru poate fi vazut ca o masura a importantei acordate valorii precedente a vitezei de deplasare pentru calculul vitezei curente. Dupa cum se va observa in continuare, alegerea unei inertii in apropierea lui 1 (rezultat ce coincide aproximativ cu algoritmul PSO) va conduce la cele mai putine experimente esuate in cazul gasirii optimului functiei utilizate.

3 Experimente si Discutii

Pentru observarea influentei inertiei asupra performantei algoritmului s-a ales ca problema de rezolvat gasirea minimului pentru functia Schaffer's f6 (figura 1), cunoscand atat forma analitca a acesteia cat si optimul global ((0,0)).

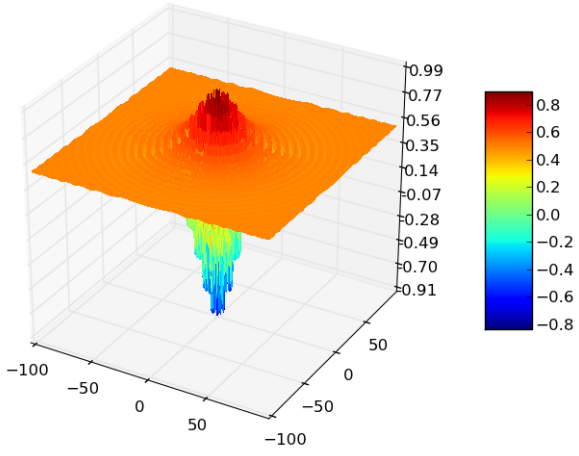

Figure 1: Functia Schaffer's f6

Programul in care au fost realizate experimentele a fost scris in limbajul de programare C++, utilizand IDE-ul CLion. Au fost realizate in total un numar de 390 de simulari (30 de initializari diferite pentru parametrii de intrare - pozitii, viteze - * 12 valori diferite pentru intertie + 30 experimente cu inertie decrementata la fiecare pas al algoritmului). Numarul total de particule pentru fiecare simulare a fost setat la 20, viteza maxima admisa pentru orice particula a fost limitata la valoarea 2 (astfel daca se ajunge, in urma unui update al vitezei, la o valoare mai mare ca 2, vitezei curente i se va atribui valoarea 2), de asemenea s-a impus o limita de spatiu pentru pozitiile fiecarei particule, si anume intervalul (-100, 100) (astfel pozitia pe orice dimensiune - in cazul functiei Schaffer's f6 doar 2 dimensiuni - va avea valori doar in acest interval, cazul depasirii intr-un sens sau altul al limitelor va fi tratat asemanator ca in cazul vitezei). Numarul maxim de iteratii admis pentru o simulare este de 4000, astfel, daca algoritmul nu reuseste sa se apropie suficient de mult de minimul functiei (diferenta dintre valoarea de minim a functiei (fitness-ul) si valoarea functiei pentru cea mai buna particule nu scade sub o anumita valoare de prag) in numarul impus de iteratii spunem ca algoritmul nu converge, simularea fiind considerata esuata.

Pentru a obesrva influenta inertiei asupra performantei algoritmului s-au realizat 30 de grupuri de experimente, fiecare fiind reprezentat de 13 experimente independente, primele 12 experimente avand asociate cate o inertie diferita pentru agoritmul prezentat, iar la ultimul s-a folosit un mecanism prin care inertia era decrementata liniar dupa fiecare iteratie. Deci, in total au fost realizate 390 de experimente. Pentru fiecare grup de experimente s-au extras dintr-o distributie uniforma U(-100, 100) valorile initiale pentru pozitiile particulelor si dintr-o distributie uniforma U(0, 2) valorile vitezelor initiale (toate experimentele din aceeasi grupa au aceiasi parametrii initiali - acestia difera doar intre oricare 2 grupuri diferite). Ultimul experiment din fiecare grup are rolul de a ilustra influenta conceptelor de explorare initiala si exploatare finala asupra performantei algoritmului. La inceput particulele sunt initializate cu parametrii random , deci prezinta o posibilitate foarte mica sa se afle in apropierea minimului, de aceea se porneste cu o valoare mare a inertiei (1.4) astfel incat sa acordam prioritate mare vitezei initiale (aleatoare), obtinand o libertate mai mare de miscare a particulei, reducand constrangerile date de pozitiile optimelor initiale (locale si global) asupra vitezei. Cu cat algoritmul evolueaza mai mult in timp se poate lua in considerare ipoteza ca importanta optimelor a depasit avantajele unui comportament aleator (de explorare a spatiului), astfel dorindu-se axarea mai mult pe informatia dobandita pana in acest moment decat adoptarea unui comportament aleator, fiind justificata scaderea inertiei de la iteratie la iteratie (evident, exista multe alte modalitati de a reduce inertia de-a lungul iteratiilor - nu doar liniara - in unele cazuri fiind chiar mai potrivita o functie de decrementare a inertiei cu o descrestere mai putin pronuntata in primele iteratii si abrupta spre final - permite un timp mai indelungat de explorare).

Fata de (1), am realizat experimente pentru inca 2 alte valori ale inertiei, si anume 1.6 si 1.8, deoarece rezultatele pentru valoarea 1.4 nu indicau vreo modificare considerabila a performantei algoritmului.

In tabelul 1 sunt prezentate rezultatele tuturor simularilor (nr de iteratii pentru fiecare experiment in parte), celulele lasate libere reprezentand experimente nereusite. Se observa ca pentru valori ale inertiei <=0.8 obtinem un numar mare de experimente esuate, in acord cu rezultatele obtinute de autorii acestei metode. Adica, in acest caz algoritmul se comporta ca o cautare mai

mult locala, tinand cont de minimele gasite de fiecare particula si de minimul global, lucru care nu are sens in contextul unor iteratii initiale in care minimele gasite pana in acest moment nu ofera prea multa informatie utila. Totusi, in cazul experimentelor pe care le-am realizat se observa ca o crestere a inertiei de la 1 la 1.4 nu afecteaza performantele algoritmului, spre deosebire de rezultatele autorilor unde numarul de experimente esuate pentru w=1.4 este 11. Astfel, pentru a observa influenta cresterii inertiei am ales inca 2 valori pentru inertie (1.6 si 1.8) caz in care se observa o scadere a performantei. Tabelul 2 prezinta numarul total de esecuri pentru fiecare inertie in parte, figura 2 fiind o reprezentare vizuala a acestor valori. Se observa ca pentru cazul in care inertia porneste de la o valoare mare (1.4) si este decrementata de-a lungul iteratiilor se obtin cele mai bune rezultate (0 esecuri si cea mai mica medie de iteratii pentru a gasi solutia), la fel ca in rezultatele obtinute de autorii articolului.
Se observa ca in intervalul [0.9, 1.4] pentru inertie se obtin cele mai bune rezultate, in timp ce autorii au obtinut cele mai bune rezultate in intervalul [0.9, 1.2]. Faptul ca am obtinut rezultate bune si pentru inertii mai mari rezulta din caracterul initial de explorare pe care il are algoritmul, nefiind influentat de minimele locale, si nici de pozitiile initiale. Totusi, generarea vitezelor initiale poate reprezenta factorul ce a determinat obtinerea unor rezultate diferite fata de autorii articolului, intrucat sunt folosite generatoare de numere aleatoare diferite.

Acestea fiind spuse, in urma desfasurarii tuturor experimentelor se observa ca am obtinut rezultate asemanatoare cu autorii articolului principal, cu o mica exceptie in cazul limitei superioare a intervalului de inertii ce conduce la performante bune ale algoritmului.

4 Concluzii

In acest raport am prezentat algoritmul PSO cat si o variatie a acestuia, MPSO, ce introduce o noua marime luata in considerare la calculul vitezei fiecarei particule, si anume inertia. Influenta noului parametru a fost observata pe un caz concret de optimizare a unei functii pentru care minimul este cunoscut, iar rezultatele au fost comparate cu cele obtinute de cei ce au introdus acest algoritm. Astfel s-a constatat ca pentru o intertie apartinand intervalului [0.9, 1.4] se obtine cea mai buna performanta, calculata ca numarul de experimente reusite cat si ca numar medii de iteratii necesare. Rezultatele obtinute coincid in mare parte cu cele obtinute de cei ce au introdus acest algoritm.

5 Referinte

[1] Y. Shi and R. Eberhart, ”A modified particle swarm optimizer,” 1998 IEEE International Conference on Evolutionary Computation Proceedings. IEEE World Congress on Computational Intelligence (Cat. No.98TH8360), Anchorage, AK, USA, 1998, pp. 69-73.

[2] Kennedy, J., and Eberhart, R. C. (1 995). :Particle Swam Optimization; Roc. IEEE International Conference on Neural Networks (Path, Australia), IEEE Service Center, Piscataway, NJ, N: 1942-1948

Nr. simulare	Inertie												Inertie variabila
	1.8	1.6	1.4	1.2	1.1	1.05	1	0.95	0.9	0.85	0.8	0	
1	821		295	487	1928	1966	1034	2065	1828	3153	3966		2141
2		2401	854	570	917	379	2631	2679	726	2448			1062
3	2739	430	635	2665	1565	1107		2860	3174	645			928
4	896	3034	2342	737	878	2055	1981	1191	2221	2388	2387		1231
5	3795	577	1988	760	2201	647	1780	797	1293	2360	1376		1225
6	392	2098	474	768	1312	1285	3643	3297	1854	308	1949		523
7		2488	2200	1107	2128	537	863	2505	1660	3038	3799		2095
8	2255	3016	1866	1520	1027	1854	2411	1963	2630				403
9	2540	2177	2021	998	1880	1037	2825	1661	937	3067	3105		1677
10	974	2479	1039	310	501	597	877	2359	709		3070		1894
11		3374	1011	589	1007	994	395	658	1487	3043	2188		902
12	431	1109	1045	271	653	229	1472	2181	376	3039			188
13	386	548	1417	580	1732	760	812	1128	591	221			622
14		1494	633	1845	1948	737	2498	1784		1006	2509		1900
15	3870	1252	2759	450	3226	2402	3537	582	1834	1945			1121
16	1833		336	993	574	605	1196	1111	450	2194	3261		1185
17	1331	2045	558	826	1471	829	1562	1424	2438	2606	1855		248
18		3235	1718	1534	576	2180	1622	708	1263	1656			1160
19	1304		459	1336	1434	1356	1023	2237	1260	2144	1987		312
20	723	2112	1640	1019	765	398	2006	2066	3555	3799	2258		651
21	3217	497	3340	1887	1961	945	2341	3429	2513	3851	1501		1213
22	426	2588	1208	2495	1519	257	1480	2326	1757	3714			461
23	2108	500	1882	1876	1078	1282	1235	1473	1236	2201	1237		1857
24	1555	991	1210	1701	278	2894	2078	3033	1507		3548		453
25	1608	783	2140	760	1450	1336	1452	3448	572	2440	2294		1773
26	425	266	479	967	482	1063	2546	533	1485	3128	3122		706
27	922	1761	1903	650	590	1428	1216	1433	1938	2277			816
28		188	167	2627	1806	624	1638	2532	2348	1241	2112		1503
29	3692		1928	1054	1267	669	1418	1224	2715	1578			421
30	1834	560	994	1060	3025	1621	2327	851	828	1654	1691	123	1556
Nr. mediu de iteratii	1669	1615	1351	1148	1372	1135	1789	1851	1627	2264	2460	123	1074

Table 1: Numarul de iteratii ale fiecarei simulari, pentru fiecare valoare diferita a intertiei si pentru cazul decrementarii inertiei dupa fiecare iteratie

Inertie	1.8	1.6	1.4	1.2	1.1	1.05	1	0.95	0.9	0.85	0.8	0
Numar de simulari esuate	6	4	0	0	0	0	1	0	1	3	10	29

Table 2: Numarul total de simulari esuate pentru fiecare inertie

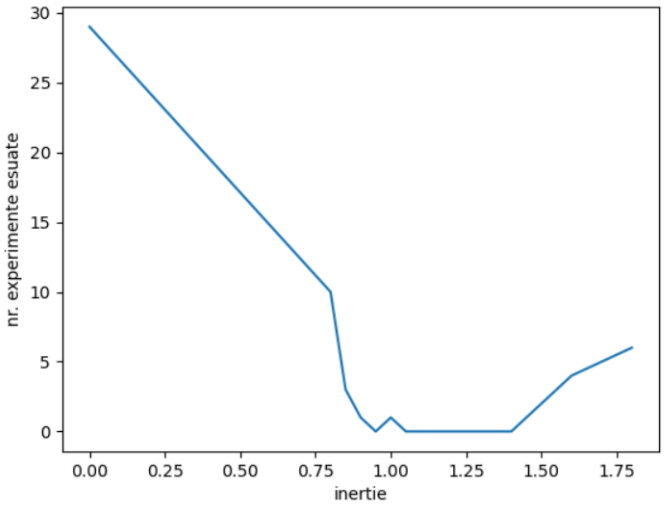

Figure 2: Numar de simulari esuate vs. inertie