Rezolvarea problemei comisului voiajor folosind un Algoritm Genetic si Simulated Annealing

Dascalu Octavian-Petrut Hamza Eduard-Mihail

January 7, 2021

1 Introducere

Problema comisului voiajor (Traveling salesman problem sau TSP) este formulata in felul urmator: "Dată fiind o listă de orase si distantele intre fiecare doua orase, care este cel mai scurt traseu posibil care viziteaza fiecare oras o singura data si se intoarce la orasul de origine?"

Cum TSP este o problem NP-HARD, prin aplicarea unui algoritm genetic si a unuia de tip Simulated Annealing se doreste gasirea solutiei problemei intr-un timp mai scurt.

Algoritmii genetici sunt tehnici adaptive de cautare euristica bazat pe principiile biologiei evolutioniste(supravietuirea celui mai bine adaptat individ), folosite in rezolvarea problemelor de optimizare, planificare sau cautare.

Simulated Annealing este o metaeuristica care aproximeaza optimul global intr-un spatiu de cautare mare pentru o problema de optimizare. Este folosit deseori atunci cand spatiul de cautare este discret, cum e cazul problemei TSP.

1.1 Motivatie

Prin acest experiment se urmareste evaluarea rezultatelor gasite din punct de vedere al medie valorilor, a valorii minime gasite si a timpului de cautare prin comparare dintre un algoritm genetic si unul de Simulated Annealing in rezolvarea problemei TSP.

2 Metode

2.1 Simulated Annealing

O solutie este reprezentata printr-un vector ce retine etichetele(numerele) celor n orase in ordinea in care se afla pe traseu.

In ceea ce priveste particularizarea algoritmului de pe pagina cursului[1], am ales sa initializez temperatura(T) cu 100, evaluarea unei solutii consta in calcularea lungimii traseului reprezentat de ea, conditia de terminare din bucla interioara este ca aceasta sa ruleze de macar 1000 de ori, temperatura T scade cu 10%, iar criteriul de terminare al buclei exterioare este ca temperatura(T) sa fi scazut sub 10^{-8}.

Un vecin random al solutiei curente este generat prin slectarea a doua pozitii random din vector si inversarea ordinii oraselor dintre cele 2 pozitii, urmata de o rotire. Pentru rotire alegem 3 pozitii random din vector, p_1, p_2, p_3, unde $p_1 < p_2 < p_3$. Vectorul initial

...	p_1	p_1+1	...	p_2-1	p_2	p_2+1	...	p_3-1	p_3	...

devine prin rotatie

...	p_2	p_2+1	...	p_3-1	p_1	p_1+1	...	p_2-1	p_3	...

.

2.2 Algoritmul Genetic

In cadrul experimentului populatia este un vetor de solutii candidat, care la randul lor sunt vectori de numere naturale ce reprezinta eticheta(numarul) unui oras de pe traseu, o solutie candidat fiind traseul parcurs de comis-voiajor.

In ceea ce priveste particularizarea algoritmului de pe pagina cursului[1], dimensiune populatiei este 100, conditia de oprire este ca t sa fie 1000(populatia sa ajunga la 1000 de generatii), iar selectia aleasa este cea de tip Roata norocului detaliata in pagina cursului[1].

Functia fitness a unei individ x din populatie este $fitness[x] = \frac{1}{lungime[x]}$, unde $lungime[x]$ este lungimea traseului reprezentat prin x.

Probabilitatea de mutatie este de 1%, iar mutatia are loc prin interschimbarea a doua orase de pe traseu.

Probabilitate de incrucisare este de 50%, incrucisarea are loc prin selectarea a 2 pozitii random, alese ca puncte de taiere, secventa de gene dintre ele fiind

transmisa copilului, dupa care se completeaza pozitiile goale din copil cu genele din celalat parinte, care nu apar in secventa de gena transmisa initial, in ordinea aparitiei.

In cadrul *while*-ului din algoritm am mai adaugat un pas in care, dupa efectuarea incrucisarii, in populatia noua se adauga populatia veche si se pastreaza cei mai buni 100 de indivizi. Dupa eliminam 5 indivizi si adaugaum 5 imigranti random in populatie.

2.3 Simulated Annealing aplicat pe rezultatul unui algoritm genetic

In urma incercarii rezolvarii problemei TSP pe mai multe versiuni de algoritmi genetici a aparut ideea aplicarii algoritmului de Simulated Annealing pe rezultatul algoritmului genetic pentru a observa daca acesta va returna rezultate mai bune.

Algoritmul genetic pe rezultatele caruia s-a aplicat Simulated Annealing este similar celui descris anterior, cu diferenta ca are probabilitatea de crossover de 30%, din populatia noua se inlocuiesc cei mai slabi 2 indivizi cu primii 2 cei mai buni din popultia anterioara, iar urmatorii cei mai slabi 2 indivizi sunt inlocuiti cu 2 imigranti random (solutii candidat generate random).

3 Experiment

Algoritmul genetic a fost rulat de 30 de ori, iar Simulated Annealing de 100 de ori pentru fiecare din instantele din tabelul de mai jos.

3.1 Algoritm genetic - rezultate

instanta	optim	best	worst	avg	σ	timp mediu(s)
berlin52	7542	8159.57	10878.5	9499.71366	574.30973	9.97
ch130	6110	14355.5	17262.3	15622.81666	729.30927	23.72
d198	15780	57753.7	70477.5	63892.53666	2900.72726	42.62
dsj1000	18659688	398961000	410439000	404328000	3404905.5139	673.74
eil76	538	688.069	895.319	770.6857	49.75349	13.72
eil101	629	977.988	1186.12	1076.1896	54.57645	19.09
kroA100	21282	36629.3	50919.6	45111.36333	3418.32893	17.84
kroA200	29368	126680	145687	136033.9	4297.60998	42.95
kroB100	22141	39887.5	52978.3	46066.96333	3708.43639	18.76
lin105	14379	29916.2	38082.3	33605.32666	2140.35636	19.38
lin 318	42029	304502	331360	314242.03333	7009.96383	81.09
rat99	1211	2162.47	2601.83	2363.275	114.4601	17.71
rd100	7910	12752	18524.2	15656.68333	1109.07835	18.56
st70	675	897.567	1196.23	1012.5097	73.33526	12.49
u574	36905	424261	450410	437085.93	6612.58943	224.99
u1060	224094	4761650	4947370	4837230	44982.84382	641.86

3.2 Simulated Annealing - rezultate

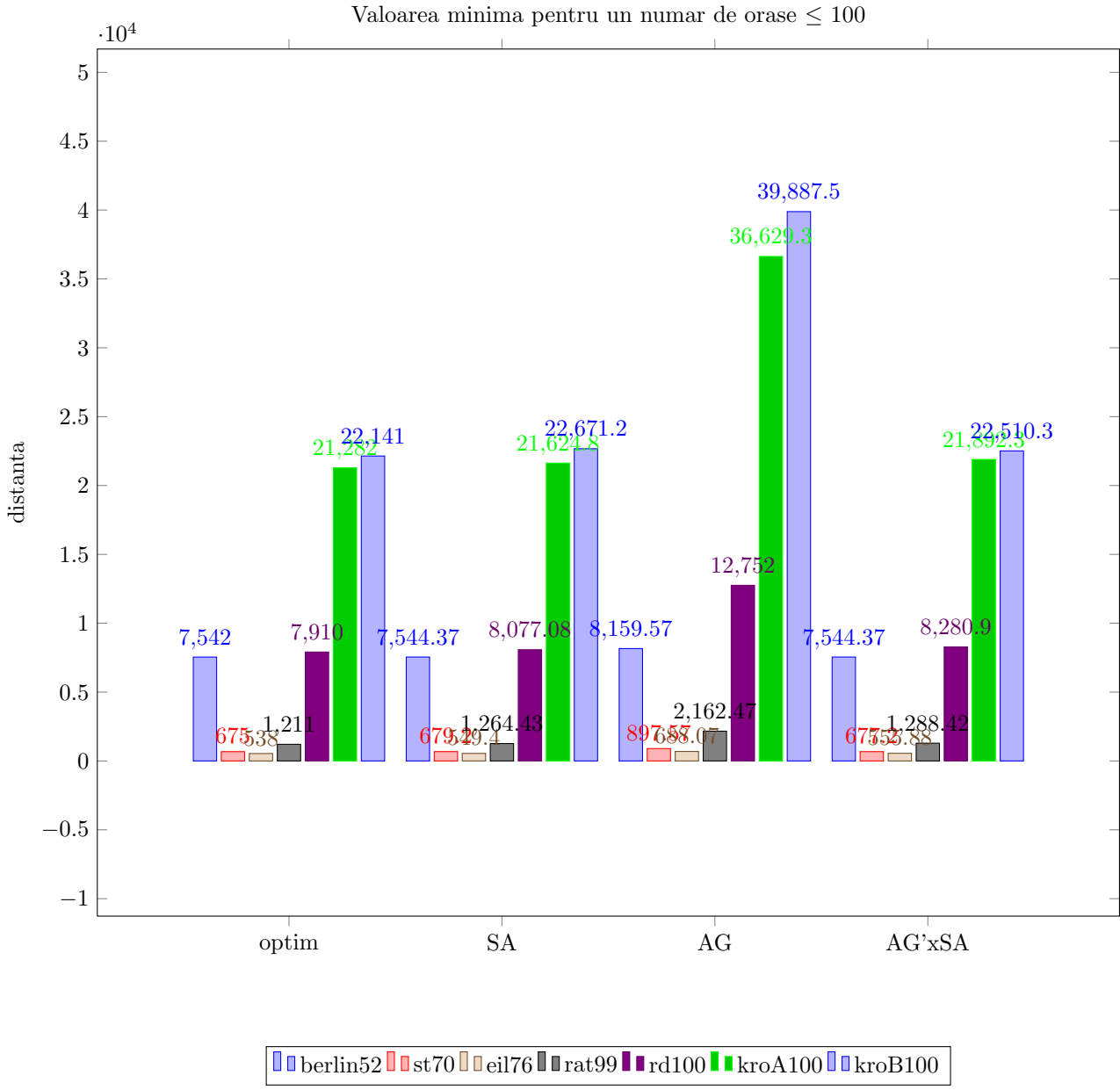
instanta	optim	best	worst	avg	σ	timp mediu(s)
berlin52	7542	7544.37	8269.66	7758.8682	205.46206	5.96
ch130	6110	6558.57	7348.46	6950.01	162.61726	10.46
d198	15780	18539.9	20023	19220.203	334.82663	14.2
dsj1000	18659688	73753000	80777500	77257343	1430066.84849	62.02
eil76	538	549.403	588.169	568.00953	7.6336	6.95
eil101	629	668.09	711.787	688.99243	9.46723	8.35
kroA100	21282	21624.8	24480	22643.812	538.35666	8.27
kroA200	29368	40795.3	41246.4	41166.504	1152.57583	14.41
kroB100	22141	22671.2	24948.4	23550.639	419.244442	8.37
lin105	14379	14751.8	16564.9	15404.442	364.35320	8.57
lin318	42029	73816.4	83484.6	78906.959	1985.86983	20.65
rat99	1211	1264.43	1415.22	1327.8711	26.92791	8.74
rd100	7910	8077.08	8414.19	8509.4279	206.20510	8.34
st70	675	679.197	725.356	697.47131	8.91475	6.69
u574	36905	99628.6	109205	104899.776	2095.47974	36.39
u1060	224094	1000330	1071920	1035600.85714	14424.07073	65.96

3.3 Algoritm genetic combinat cu Simulated Annealing - rezultate

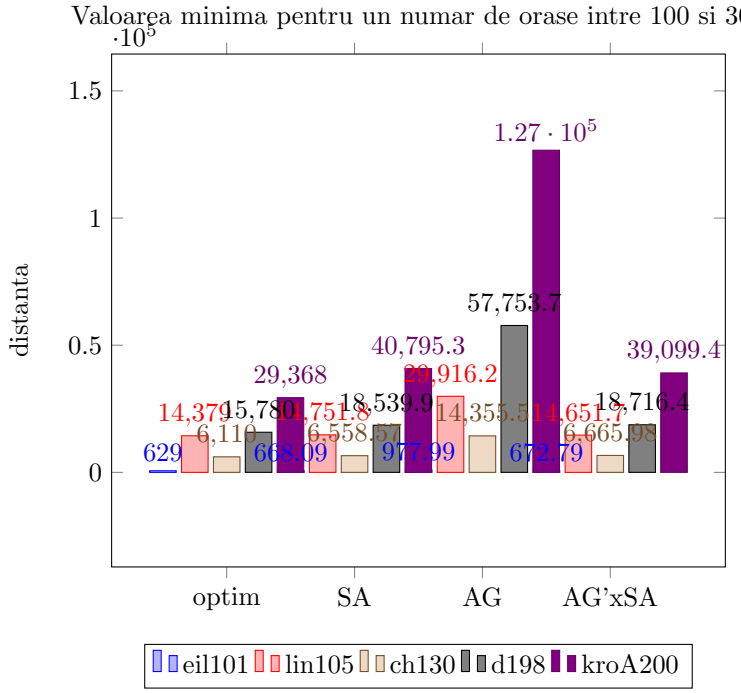
instanta	optim	best	worst	avg	σ	timp mediu(s)
berlin52	7542	7544.37	8171.82	7789.55166	196.41803	12.32
ch130	6110	6665.98	7259.18	6954.082	145.28819	27.04
d198	15780	18716.4	19946.9	19321.52	342.63617	42.86473
dsj1000	18659688	73848800	79958700	76770116.66666	1349280.32901	488.09
eil76	538	555.882	585.082	570.26306	6.67476	17.12
eil101	629	672.79	711.247	690.14876	10.38123	21.2
kroA100	21282	21892.3	23752.4	22762.31666	500.3324	20.68
kroA200	29368	39099.4	42998.8	40781.94666	970.01125	43.43
kroB100	22141	22510.3	24092	23427.27666	425.82077	20.68
lin105	14379	14651.7	16316.3	15348.56666	403.77825	21.59
lin318	42029	75145.6	81262.7	78649.08333	1631.97851	82.76
rat99	1211	1288.42	1380.98	1323.21433	22.06192	20.52
rd100	7910	8280.9	8979.13	8523.97733	186.02543	20.74
st70	675	677.195	712.864	692.3369	7.81522	15.29
u574	36905	101242	107968	104650.73333	1943.49677	201.14
u1060	224094	989518	1070400	1037000.6	15730.22755	544.39

4 Rezultate

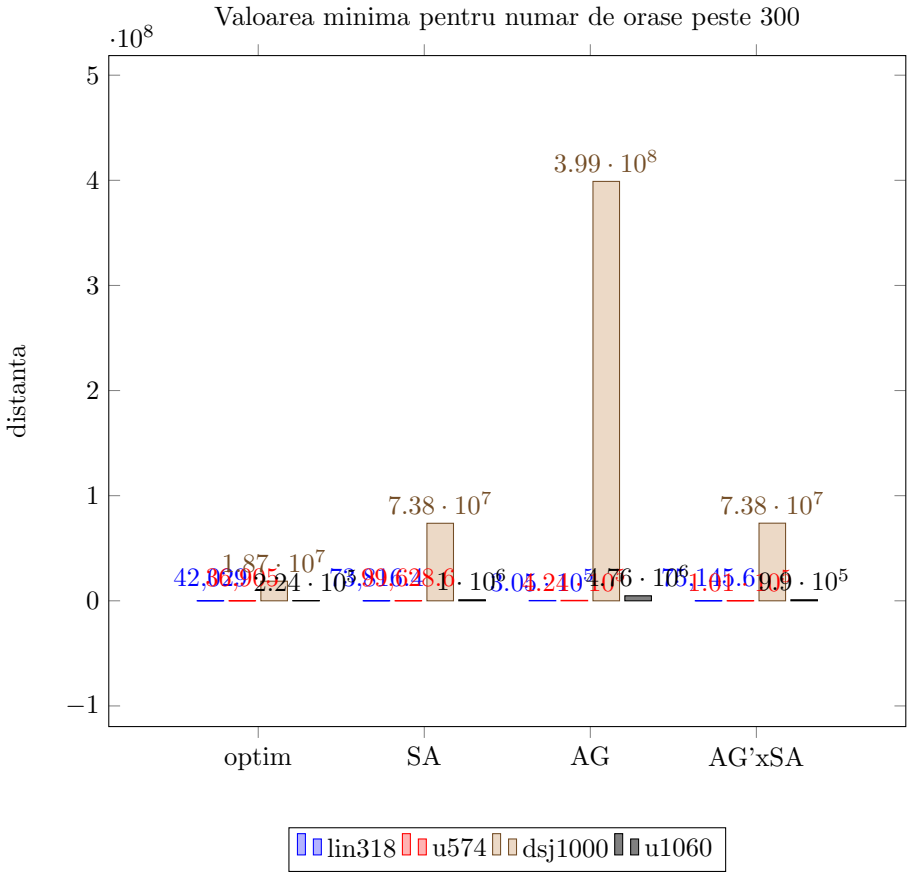
4.1 Valoarea minima a solutiei gasite



Pentru instante cu numar de orase mai mic de 100 Simulated Annealing gaseste valoarea minima cea mai apropiata de optim.

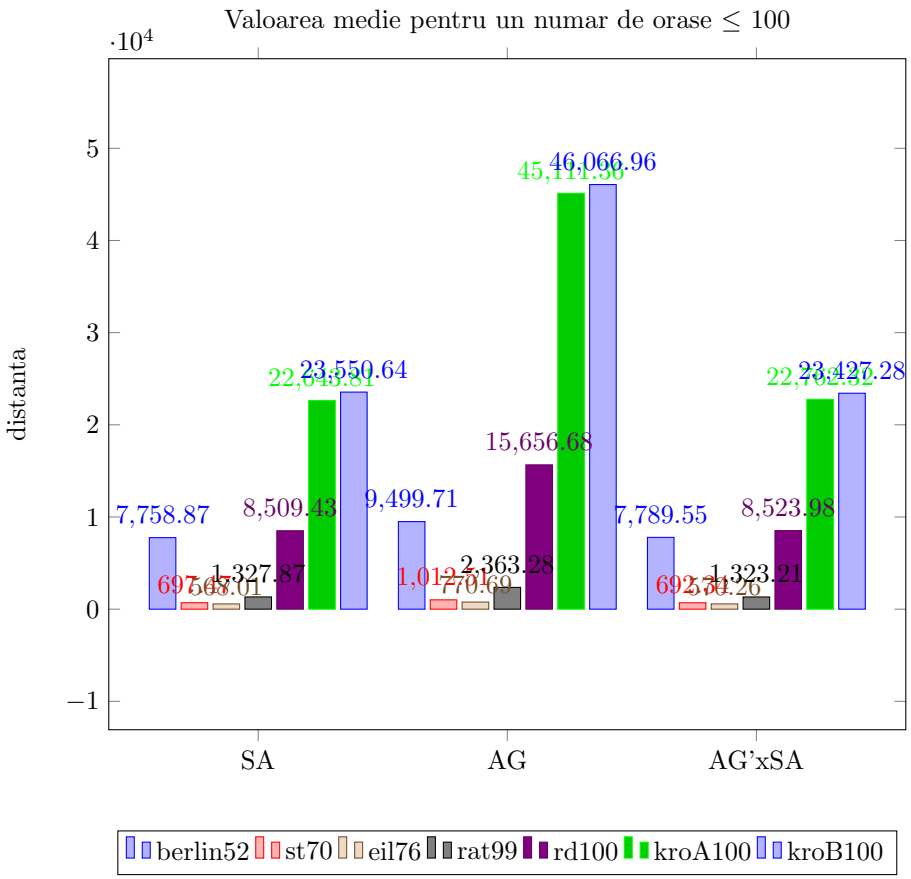


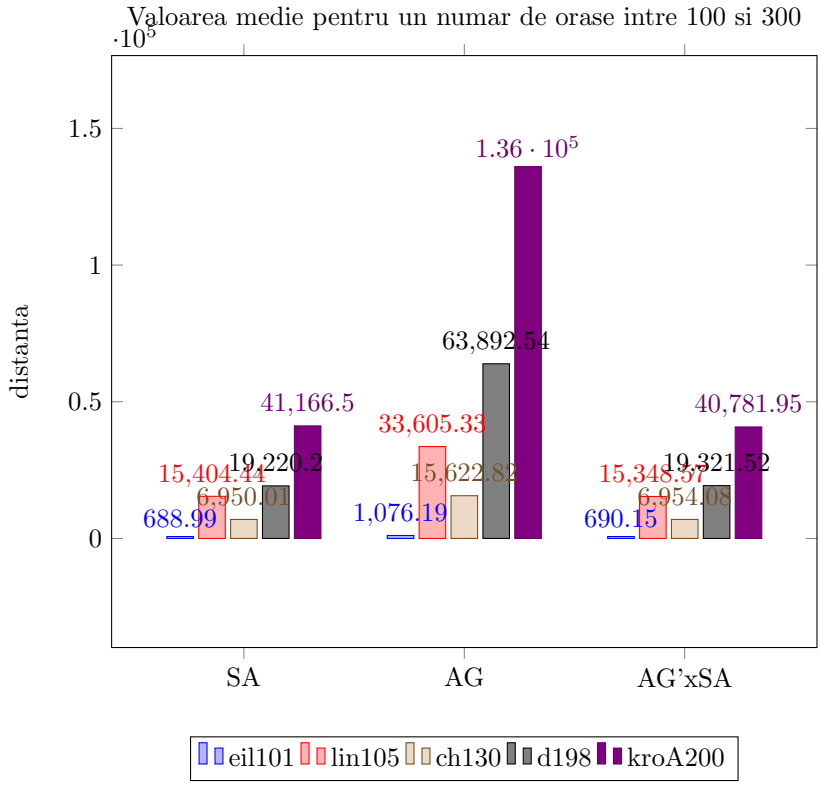
Pentru instante cu numar de orase intre 100 si 300 Simulated Annealing gaseste valoarea minima cea mai apropiata de optim tot timpul, insa incepand cu 130 de orase diferenta dintre optim si aceasta devine din ce in ce mai evidenta.

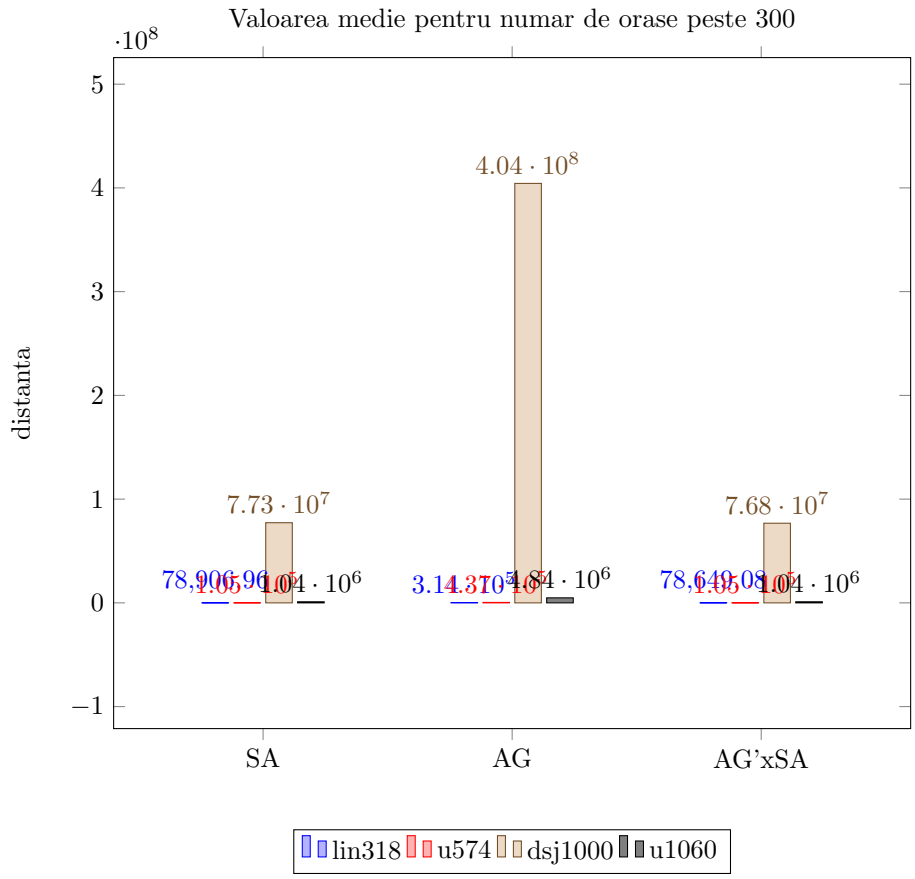


Pentru instante cu numar de orase peste 300 Simulated Annealing gaseste valoarea minima cea mai apropiata de optim tot timpul, diferenta dintre optim si aceasta devine din ce in ce mai evidenta cu cresterea numarului de orase.

4.2 Valoarea medie a solutiei gasite

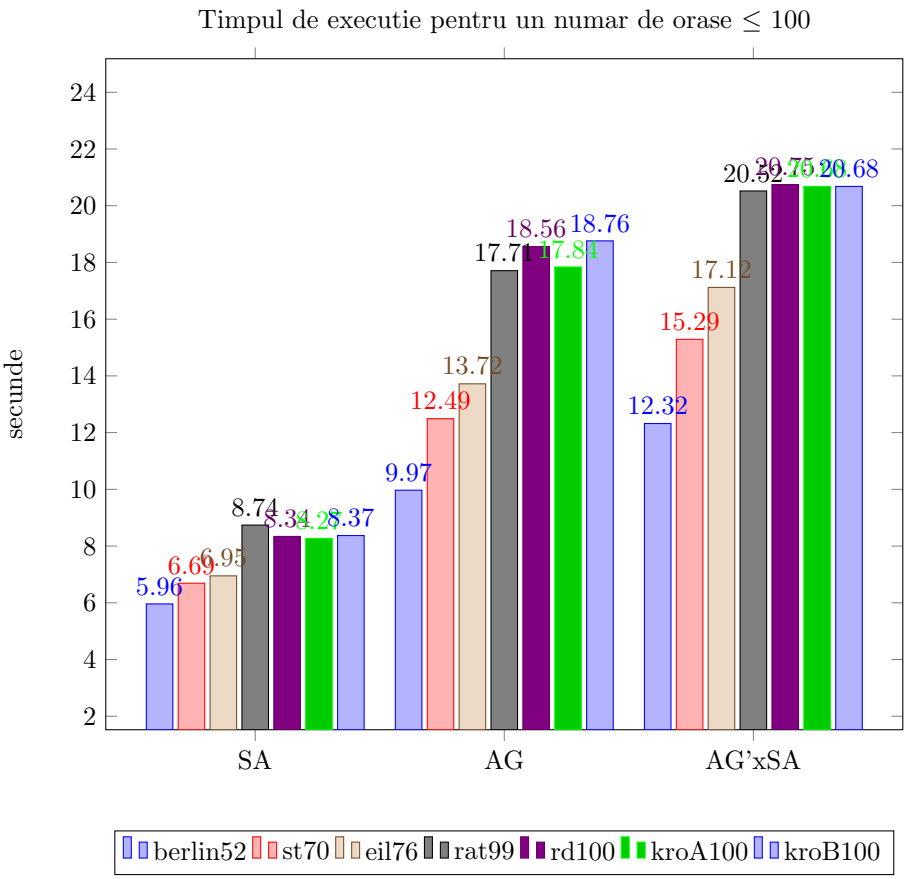



Valoarea medie pentru un numar de orase intre 100 si 300
$\cdot 10^5$
distanta
1.5
1
0.5
0
$1.36 \cdot 10^5$
63,892.54
41,166.5
33,605.33
15,404.44
19,220.2
15,622.82
6,950.01
688.99
1,076.19
15,348.57
40,781.95
19,321.52
6,954.08
690.15
SA
AG
AG'xSA
eil101
lin105
ch130
d198
kroA200

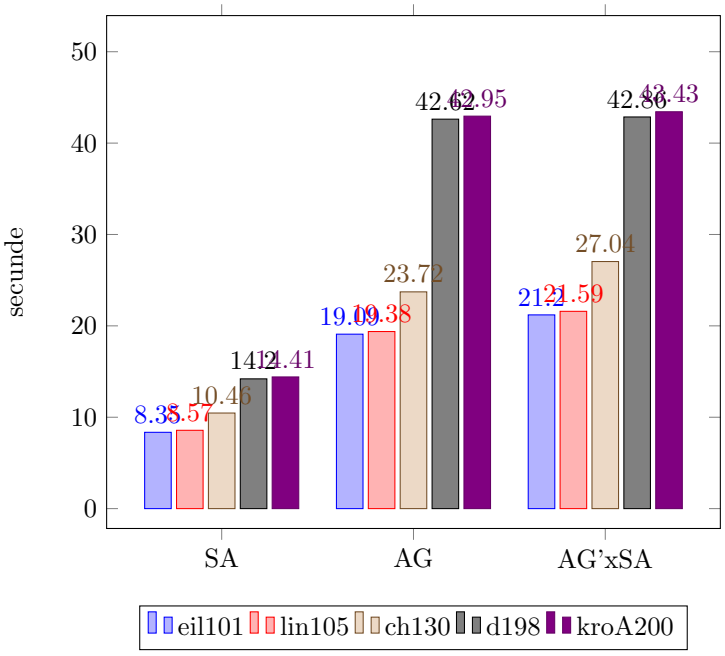


Cea mai buna valoare medie se obtine pentru Simulated Annealing, urmata la diferenta mica de Algoritmul gentic incrucisat cu un Simulted Annealing.

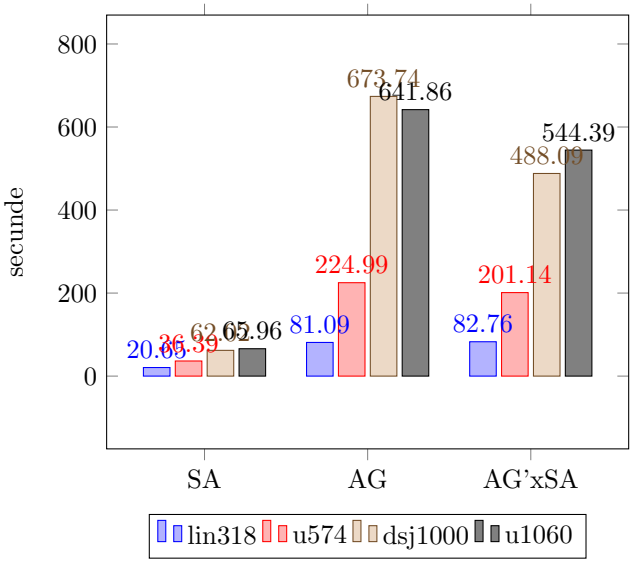
4.3 Timpul mediu de cautare



Timpul de executie pentru un numar de orase intre 100 si 300



Timpul de executie pentru numar de orase peste 300



Durata medie a rularii unui Simulated Annealing este mai mica decat cea

a unui algoritm genetic, pentru instantele mici, iar cei 2 algoritmi genetici au valori de timp similare.

5 Concluzii

In urma rezultatelor din experiment se poate trage concluzia ca in cautarea solutiei TSP pentru distante mici, sub 130, este de preferat utilizarea Simulated Annealing-ului, datorita valorii apropiate de optim si a timpului scurt. Cum cu cresterea numarului de orase valoarea Simulated Annealing-ului devine din ce in ce mai slaba, ar trebui sa alegem utilizarea unui algoritm genetic optimizat fata de cel prezentat mai sus(acesta avand valori slabe), care ar reusi sa gaseasca un drum mai bun. In ceea ce priveste incrucisarea unui Algoritm genetic, cu rezultate slabe, cu un Simulated Annealing, rezultatele sunt mai bune decat cele originale, dar nu decat cele pentru Simulated Annealing.

References

[1] Pagina cursului de Algoritmi genetici
`https://profs.info.uaic.ro/~eugennc/teaching/ga/`

[2] Distributie uniforma nr. reale
`https://en.cppreference.com/w/cpp/numeric/random/uniform_real_distribution`

[3] Ditributie uniforma nr. intregi
`https://en.cppreference.com/w/cpp/numeric/random/uniform_int_distribution`

[4] Simulated Annealing - Wikipedia
`https://en.wikipedia.org/wiki/Simulated_annealing`

[5] TSP - Wikipedia
`https://en.wikipedia.org/wiki/Travelling_salesman_problem`

[6] Instante TSP
`http://comopt.ifi.uni-heidelberg.de/software/TSPLIB95/tsp/`

[7] Algoritmi genetici - Wikipedia
`https://en.wikipedia.org/wiki/Genetic_algorithm`

[8] Generarea unei permutari de la 1 la n folosind algoritmul Fisher–Yates
`https://www.geeksforgeeks.org/generate-a-random-permutation-of-1-to-n/`

[9] Creare vecini solutie - Simulated Annealing
`https://www.technical-recipes.com/2012/c-implementation-of-hill-climbing-and-simulated-annealing-applied-to-travelling-salesman`

[10] S.N.Sivanandam, S.N.Deepa. *Introduction to Genetic Algorithms*. Springer, 2008.