Optimization Algorithm Comparision

Report By: Prakhar Gurawa, 20231064

Problem Description:

We have considered the fruit picking problem for this assignment, where there are n trees (n = 10) which are spread over 2-Dimensional space each represented using x coordinate, y coordinate, and a number of fruits on that tree. We need to place 6 baskets (12 real-valued decision variables) so that the amount of work done is minimum.

$$\textbf{Objective function = Amount of work done = Minimize } f(x_1,x_2,....,x_{2m}) = \Sigma_i\ w_i\ \min_j(d(t_i,b_j))$$

Our objective function is actually the cost function, which we need to minimize. So it's a **minimization problem**.

Choice of algorithms :

We have considered the following algorithms with variations of there hyperparameters such as Hill Climbing HC, Simulated Annealing SA (Initial temperature T, Temp. decay rate alpha), Late Acceptance Hill Climbing LAHC (History length L), Genetic algorithm GA (Population Size P, Number of generations G, Mutation Ratio M, Tournament size T), Covariance Matrix Adaption CMA (Population size P) and finally Particle swarm Optimization PSO (Particle velocity scaling factor W, Pbest scaling factor CP, Gbest scaling factor CG).

Also, we have limited our fitness evaluation budget to 50,000 for each algorithm and for population-based algorithms budget = popsize*ngens.

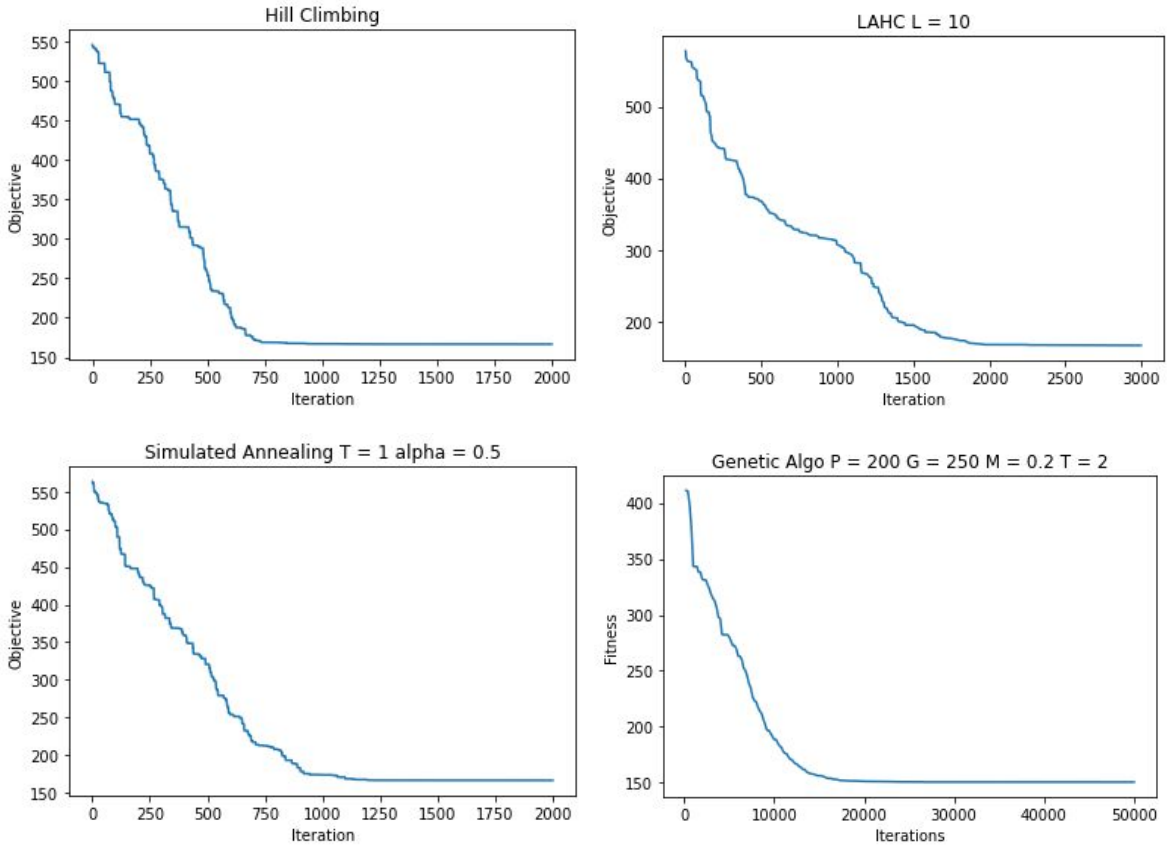
Experimental Design:

For this experiment, we have considered the standard factorial design which varies the number of hyperparameters for each optimization algorithm. Below is the table of observation:

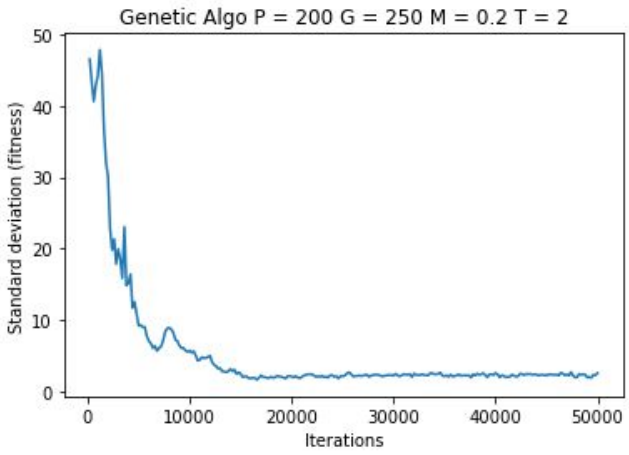
Number	Algorithm	Param 1	Param 2	Param 3	Param 4	Obj. Mean	Obj. Std
1	HC	None	None	None	None	166.472992089	0.00144282126
2	SA	T=1	alpha=0.5	None	None	160.873341994	6.97426134480
3	SA	T=1	alpha=0.9	None	None	149.675299433	15.2620742312
4	SA	T=2	alpha=0.5	None	None	163.274173077	6.40058763596
5	SA	T=2	alpha=0.9	None	None	139.273875228	3.91926213101
6	LAHC	L=2	None	None	None	142.474381190	5.05821926105
7	LAHC	L=10	None	None	None	128.387331119	31.3675077547
8	LAHC	L=20	None	None	None	132.165396314	15.1839485588
9	GA	P=100	G=500	M=0.1	T=2	157.100584968	7.82689336872
10	GA	P=100	G=500	M=0.2	T=2	150.541201248	0.02633051417

11	GA	P=200	G=250	M=0.1	T=2	150.720782980	0.09180058409
12	GA	P=200	G=250	M=0.2	T=2	149.410270353	2.28201608506
13	CMA	P=10	None	None	None	96.8498447256	13.3346914426
14	CMA	P=100	None	None	None	70.6832815730	5.36656314596
15	CMA	P=200	None	None	None	68.0000000000	7.0485839e-5
16	PSO	W=0.5	CP=0.5	CG=0.5	None	82.4618125553	11.1715928550
17	PSO	W=0.5	CP=0.5	CG=0.9	None	75.1610568684	7.06410655617
18	PSO	W=0.5	CP=0.9	CG=0.5	None	74.1301686804	6.62400698337
19	PSO	W=0.5	CP=0.9	CG=0.9	None	77.9993539430	4.25961091501
20	PSO	W=1	CP=0.5	CG=0.5	None	76.1540917550	5.87013540415
21	PSO	W=1	CP=0.5	CG=0.9	None	71.4149603427	3.46251961388

Plots(Objective Function vs Iteration):



Plot (Objective Function standard deviation vs Iteration):



Results/Observations:

1. After this experiment, we can observe that **CMA performs best** among all other algorithms with a mean objective value of 68 and a minimal standard deviation of 7.0485839e-5 which makes it a very stable algorithm.
We have also observed that **PSO performs on par with CMA** and better than others with a mean objective value of 71.41496 for configuration W=1, CP=0.5, and CG=0.9.
2. The nature of the search landscape seems to be **multimodal with some little noise**. The above objective function has multiple optimal solutions.
3. For population-based algorithm keeping the budget fixed, the **algorithm performs well when the population is increased**. We can observe this in the case of G=100 objective values comes out to be 157 and 150, whereas for G=200 it is 150 and 149. (GA Algorithm). We can further test using some higher values of population
4. By the above objective function vs iteration graphs we can clearly infer that algorithms are getting converge **nearby 2000 iterations**. So further iterations will be of no use as the algorithm will be highly **exploitative** and not explorative.

Code Structure:

Main class OptimizationAssignment2.py which imports all other individual algorithms.