IT3105 - Artificial Intelligence Programming

Particle Swarm Optimization
Implementation of a PSO algorithm

Marien Fressinaud
marienlf@stud.ntnu.no

24th November 2013

Table of Contents

1.Introduction....................2
2.The circle problem....................3
Task 1....................3
1D Circle problem....................3
2D Circle problem....................4
Task 2....................5
Nearest neighbour topology....................5
Weight of inertia decreasing....................6
3.The 0-1 Knapsack problem....................7
Task 3....................7
Basic problem....................7
Weight of inertia decreasing....................8
Task 4....................9

1. Introduction

Particle Swarm Optimization (PSO) is an algorithm which is inspired by living organisms such as birds or fishes. The idea is to move several particles in a search space (each representing a solution to a given problem) according to both velocity and position. Particle movement is influenced by its inertia and the best personal and swarm known positions.

This report presents the results of my own implemented algorithm of PSO. Each part uses graphs and / or tables for presentation and tries to draw a conclusion using the results.

The source code has been written in Python relying on its clear syntax. The different tasks have been seperated in different directories but source code is always quite similar:

1. First, we declare constants like the max number of iterations, limits of velocity or the number of particles.
2. Then, there is declaration of the function to evaluate
3. After, a Particle class describes functionment of a particle: initialisation, evaluation of the function, update of velocity and position
4. Finally, the fourth part describes swarm mechanisms: generation, evaluation of all the particles, calculation of the best fitness and update of positions.

2. The circle problem

The circle problem is defined as: $f(u_1,\dots,u_n)=(u_1*u_1)+\dots+(u_n*u_n)$ and we should minimize f. An obvious answer is to take $u_i=0$ with $i\in[1,n]$.

For this problem, we took 10 particles to resolve problems.

Task 1

1D Circle problem

In the following graph, we can see movements of 10 particles (each with a different color). We observe that all of them are approaching the value 0. Algorithm is stopped when the fitness is less or equal 0.001.

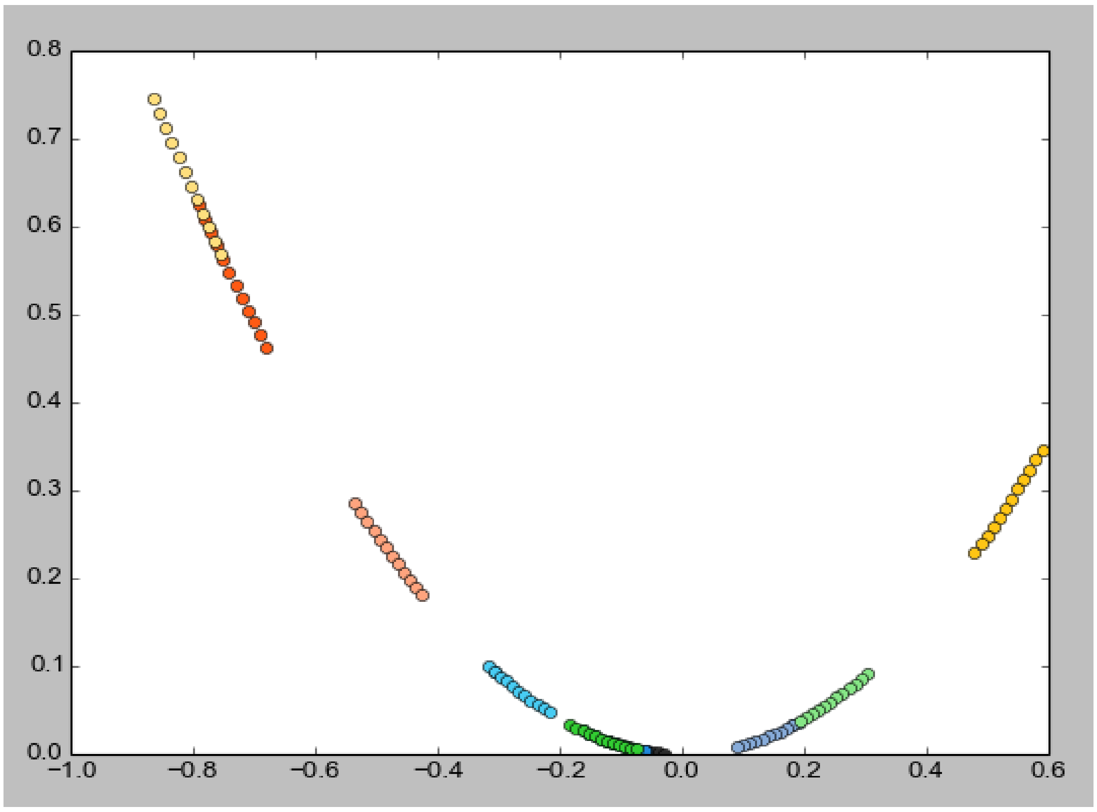


Illustration 1: a graph illustrating search of a solution for 1D circle problem

The following table gives results of 10 runs:

Run	1	**2**	3	4	5	**6**	7	8	9	10
Fitness	0.001	**0.000**	0.001	0.001	0.001	**0.001**	0.001	0.001	0.001	0.001
Iteration	31	**2**	11	8	16	**2**	6	15	22	13

Algorithm always gives a solution in less than 40 tries which is quite good. By observing extremities, we see very different results. In fact, initial position of particles has a big impact at this level.

2D Circle problem

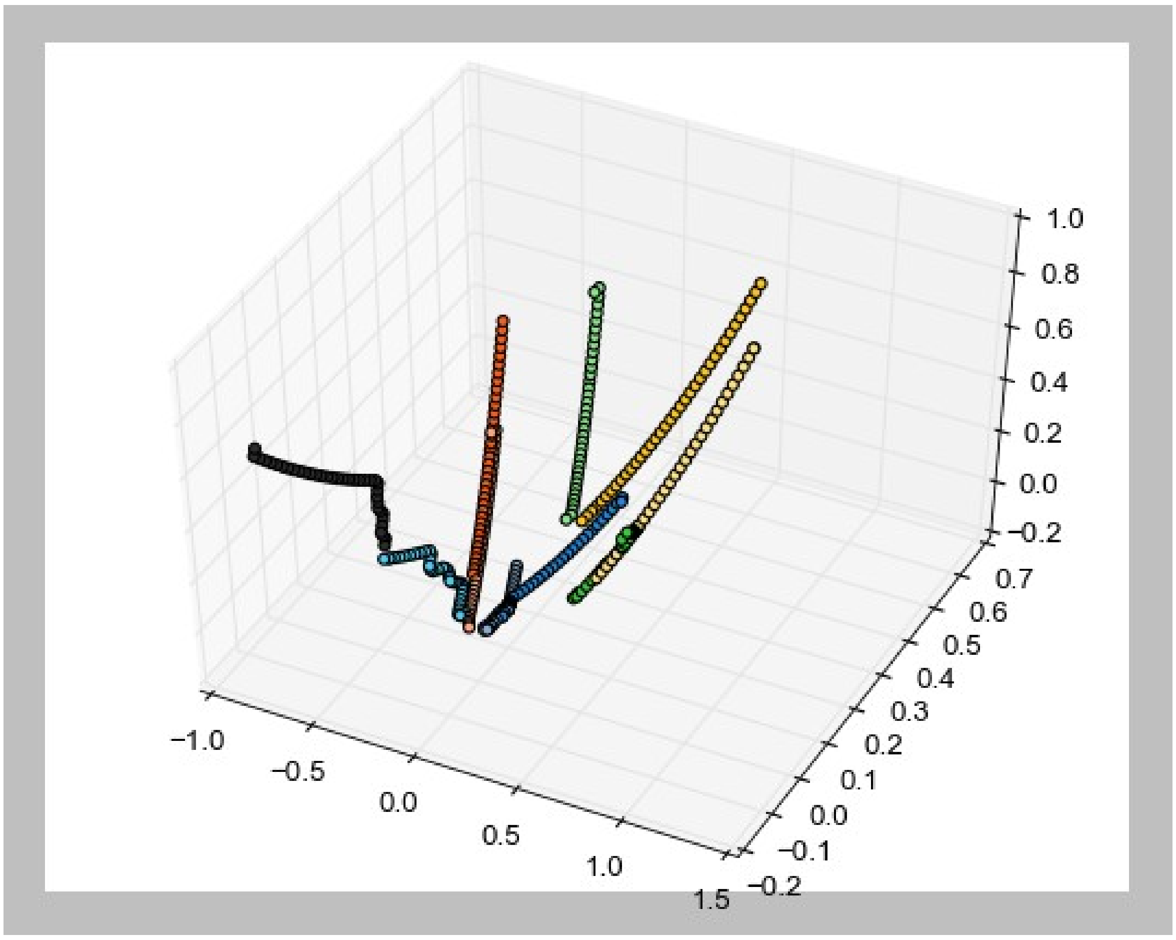


Illustration 2: a graph illustrating search of a solution for 2D circle problem

Run	1	2	3	4	5	**6**	7	8	9	10
Fitness	0.001	0.001	0.001	0.001	0.001	**0.000**	0.001	0.001	0.001	0.001
Iteration	59	56	39	57	43	**34**	44	62	66	43

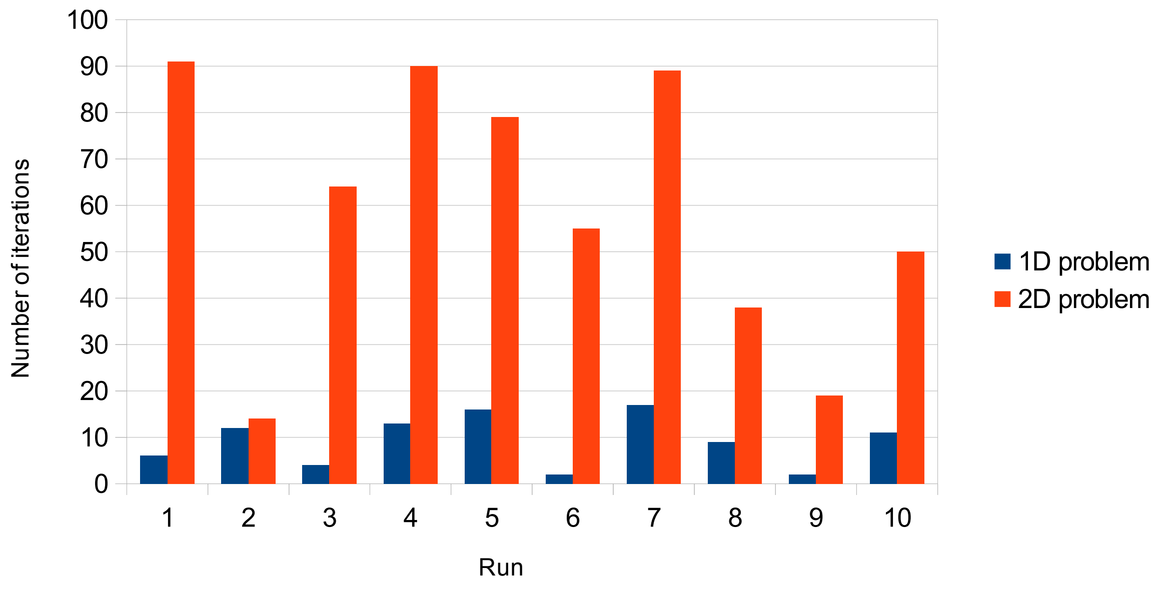
For this problem, finding a correct answer is a bit more complicated, but it is still under 100 tries and a solution is found for each run.

It is normal to spend more time to find a correct solution since particles are moving in a 2-dimensions search space.

Task 2

Nearest neighbour topology

The following graph compares results for 1D and 2D problems. The fitness is not representing since it was always less or equal 0.001.

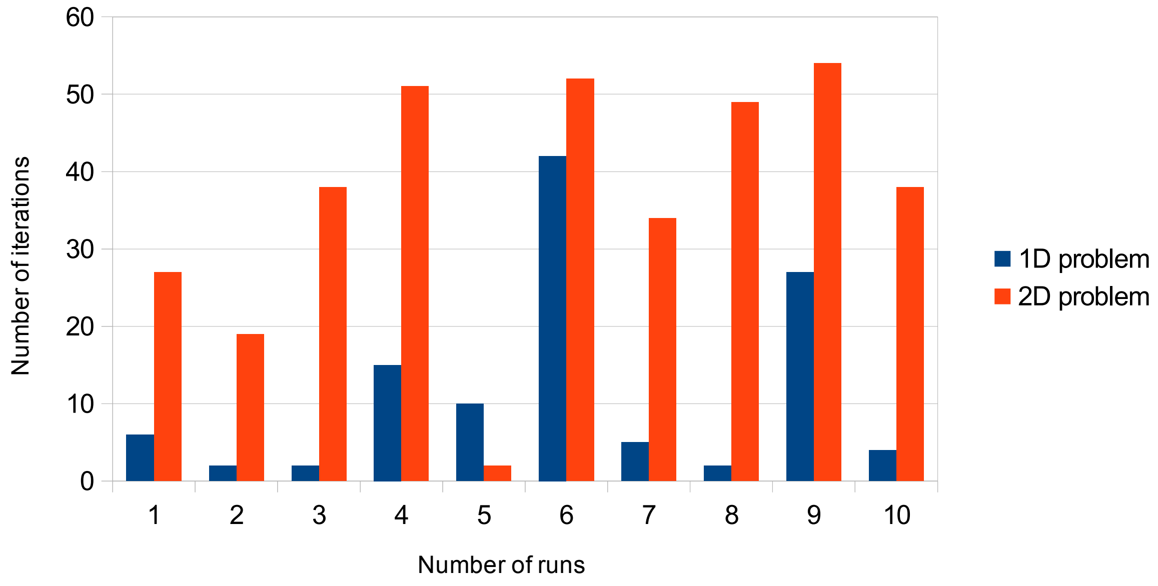


Results for 1D problem are still good. Since the particles are evolving in only one dimension, they don't have many choices to move and it's normal to have similar results than previously. We have also observed good results with only 2 particles in the swarm or by considering only one neighbour.

However, 2D problem is more interesting. Now, metrics show more bad results. Using knowledge of a maximum of the neighbours seems essential when the number of dimensions are increasing. Indeed, particles will have more and more possibilities to move. Considering knowledge of all the particles will "decreasing" the number of possibilities (choice will be more reliable).

Weight of inertia decreasing

By decreasing the weight of particle inertia, we limit possibilities to explore search space. Indeed, particles will be focus on existing knowledge even if it is far from a good solution. Idea is to decrease the weight slowly in order to be able to explore search space a maximum at the beginning and then considering only the best positions.



We observe the results for 1D are generally better than those of previous task, except for the runs number 6 and 9.

But the 2D problem illustrates very well aim and problem of decreasing the inertia weight. Results are currently better than the previous ones (never more than 60 iterations). But during development, with a more drastic decreasing function (e.g weight was decreasing by 0.95 at each iteration), it was current to not find a good solution and we reached quite often the 1000 iterations limit.

Decreasing weight is a two-edged sword and should be used with precaution.

3. The 0-1 Knapsack problem

Due to performance issues, only 50 particles were used in the search space for this problem. Nevertheless, results are good enough.

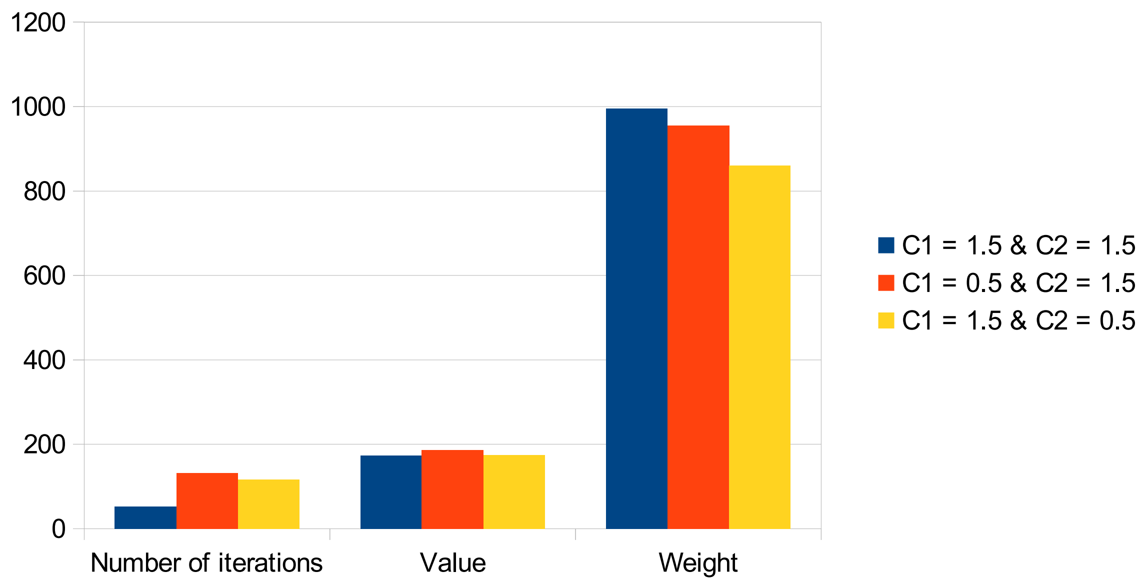
Task 3

For this problem, time has been measured. It was variyng between 15 minutes (only algorithm was running) and 30 minutes (a web browser and an email platform were used at the same time).

Graphs presenting in this part compare three metrics for different values of C1 and C2:

- Number of iterations before finding the best current solution
- The value of the best found solution
- The weight of the best found solution

Basic problem

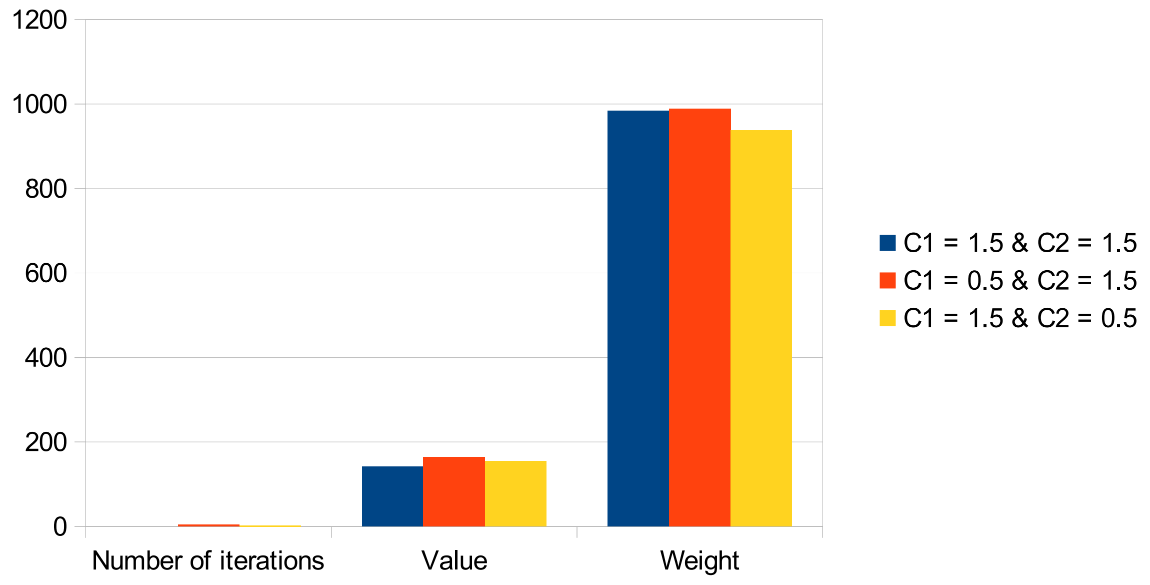


Values are quite similar and good for all the cases (around 180, 200 seems to not be reachable). So C1 and C2 seem have no impact on the final result.

Weight is less interesting since it is always around 900 (where 1000 is the maximum and the average weight is 50). Nevertheless, we can observe for C2 = 0.5 that we were certainly be able to find a better solution since we are quite far from the limit. We can conclude the weight given to the group influense is really important to help particles to improve the current best solution.

Number of iterations is more interesting: by giving more weight to C1 and C2 than inertia, number of iterations needed to find the best solution is lower. In all the cases, stopping algorithm at 200 iterations would be good enough.

Weight of inertia decreasing



There are three things we can note here:

- Number of iterations needed to find the best solution is really low.
- Best values are lower than previous tests (around 150)
- Weight values are all near the maximum

That confirms our previous assumptions: weight of inertia is required to explore the search space. By decreasing it, we can improve current best solution: actually we can not really add new packages to our solutions.

Current results show the weight was decreasing too quickly and we were not be able to find better solutions after that.

Task 4

This task took 40 minutes to finish. We chose C1 = 1.5, C2 = 1.5 and 50 particles as earlier. Volume for each package has been randomly taken uniformely between 1 and 100.

	Best solution
Number of iterations:	397
Value:	148.10
Weight:	901.65 (max 1000)
Volume:	912.57 (max 1000)

Result is not surprising and show a quite good solution: there is no many more possibilities to add a new package without violating one of the two constraints (weight and volume limits).