Group 1

Application of Particle Swarm Optimization Algorithm in the Heating System Planning Problem

Name: Muhammad Ikhwan Haqim Bin Mohd Rizam
Matric Num: B022010076
Section: 3 BENR S2
Email: B022010076@student.utem.edu.my

Name: Muhammad Khairin Nabil Bin Rozidan
Matric Num: B021910027
Section: 3 BENR S2
Email: B021910027@student.utem.edu.my

Name: Elias A/L Edward
Matric Num: B022010113
Section: 3 BENR S2
Email: B022010113@student.utem.edu.my

Name: Muhammad Naufal Bin Md Khir
Matric Num: B022010046
Section: 3 BENR S2
Email:B022010046@student.utem.edu.my

Abstract— In this study, we discuss the challenge of designing heating systems by using the Particle Swarm Optimization (PSO) method to the problem. Although this kind of heating system is not particularly common in Malaysia, it is highly common in countries with four different seasons of weather; however, for the purpose of this discussion, we will focus on China. The overall goal of this research is to bring down the price of heating systems. Our main plan for researching this case is research conducted on the Internet. The findings of this case study demonstrate the usefulness of PSO in improving the efficiency of planning procedures for heating systems. We modify it so that if PSO is used in the right way, we will be able to see how successful this technique is in solving the issue with the heating system.

Keywords—particle swarm optimization (PSO), heating system planning

I. INTRODUCTION

The phrase "climate change" refers to long - term changes in temperature and weather systems that occur over longer periods of time. Modifications like this may have been brought on by natural causes like changes in the sun's cycle, for instance. Since the 1800s, the primary factor responsible for climate change has been activities carried out by humans, in particular the use of energy sources that are fossil fuels like coal, oil, and gas for power generation. The burning of fossil fuels results in the release of greenhouse gases. These gases act as a covering over the Earth, causing temperatures to rise as more heat is trapped inside layers.

Greenhouse gas emissions such as carbon dioxide and methane, are two examples of the factors that lead to the global climate change. These are produced when activities like as driving a vehicle with gasoline or using coal to heat a building are carried out as samples. The release of methane gas from landfills is a significant contributor to environmental pollution. Energy, industry, transportation, buildings, agriculture, and land use are some of the major contributors to greenhouse gas emissions [1].

For the sake of our case study, we will be concentrating on China, which is in the midst of significant growth and development in many of its transportation-related industries. As a direct outcome of this, both the amount of energy consumed and the release of greenhouse gases will increase [2].

As a result of the completion of the systematic restructuring, the adjustment of China's energy structure, and the demand for environmental protection, the heating energy structure has been changing, and it has been encouraging the growth of heating mode. This is due to the fact that the heating mode has been stimulating the growth of heating mode. It is very necessary to carefully research, analyze, and choose the mode of heating that is perfectly suited to the characteristics of the location.

As the rate of development sped up, there will be an increase in the number of heating systems installed. This is because metropolitan areas absolutely need infrastructure. It is important to do research on the most successful architecture for a heating system in order to bring down the overall cost of the project and the amount of energy required for heating.

The main objective of this research is to investigate the potential applications of the PSO algorithm in the process of reducing the cost of heating systems, which is the issue that has to be addressed when developing heating systems. In conclusion, the findings of this case study demonstrate the value of PSO in maximizing the efficiency of planning procedures for heating systems.

II. THEORY

The Particle Swarm Optimization (PSO) method is one of the most widely used methods in the field of computational intelligence, especially in the fields of artificial intelligence, engineering design, and image processing [3]. Particle Swarm Optimization, generally known as PSO, is a population-based probabilistic algorithm model that Kennedy and Eberhart (1995) suggested using swarming birds as an example. This model is based on the social behaviors of birds. It was a stochastic algorithm that was based on a population [4].

In its most fundamental form, optimization refers to the technique that finds the highest or lowest possible value of an objective function or process [5]. The primary purpose of optimization is to find the values of the variables that minimize the objective function or provide the "best" design relative while fulfilling the constraints. To put it another way, we want to find the best possible solution out of all the options that are available. These include, among other things, optimizing productivity; maximizing strength; improving dependability; optimizing lifetime; optimizing efficiency; and optimizing consumption [6].

In general, there are four different categories of optimizations. They are deterministic optimization, continuous optimization vs discrete optimization, unconstrained optimization against discrete optimization, none, one or many goals, and continuous optimization versus discrete optimization. If we compare discrete optimization issues to continuous optimization problems, we can see that discrete optimization problems are more difficult to solve. When compared to limited optimization, unconstrained optimization issues are more usual and in the new formulation of constrained optimization problems. This is because unconstrained optimization allows for more freedom in problem solving.

When the variables are clearly limited, it might be difficult to find an optimal solution using unconstrained optimization. Constrained optimization problems may be further classified according to the nature of the constraints, which can be linear, nonlinear, or convex, as well as according to the functional smoothness, which can be differentiable or not differentiable. When there are several goal optimization issues happening in streams, one, none, or all of the objectives may often be used as possible solutions. In some contexts, it is also referred to as a single-objective issue [7]. Deterministic optimization is the third and final kind of optimization, which competes with stochastic optimization. A condition is considered to be deterministic optimized if it has the following characteristics: accurate input; accurate starting values; accurate measurement; and accurate deterministic optimization. As the following equation shows, this deterministic optimization is very dependent on linear algebra.

$$\mathbf{J^* = J\ (x^*, u^*)}$$

In stochastic optimization, the state is characterized by a known dynamic process and an unknown input, in addition to an incomplete initial condition and an imperfect or partial measurement. This is because stochastic optimization uses an incremental algorithm. A probability distribution was used to specify the data for this stochastic optimization, which results in the data having a degree of uncertainty. The equation for stochastic optimization is as follows:

$$\textbf{Optimal cost} = \mathbf{E\ \{J\ [x^*, u^*]\}}$$

The particle swarm optimization (PSO) method begins with the formation of a group of random particles, each of which represents a solution to the problem. In order to determine the optimal value, each particle performs an update on the generation process, also known as iteration. At each cycle, each particle receives an update that is based on the two most recent values. The basic particle swarm optimization takes use, at the t-th iteration, of both the current best g* for the whole population as well as the best x* for each individual. In despite the fact that a certain amount of chance may be used to simulate this variety, one of the main reasons to employ individuals is to raise the number of outstanding alternatives that are offered. As a consequence of this, there is no strong reason to choose the individual solution that is greatest [8], unless the optimization problem in question is highly nonlinear and multimodal.

The following equation may be used to get an updated position for the particles:

$$\mathbf{x^{t+1} = x^t + v^t}$$

where:

x^t : position of particle i at iteration

tv^t : velocity for each particle

The equation also updates velocities at the same time:

$$\mathbf{V^i_{k+1} = wv^i_k + c_1r_1(xBest^t_i - x^t_i) + c_2r_2(gBest^t_i - x^t_i)}$$

Where:

r1 and r2: random numbers (0-1)

c1 and c2: positive constants

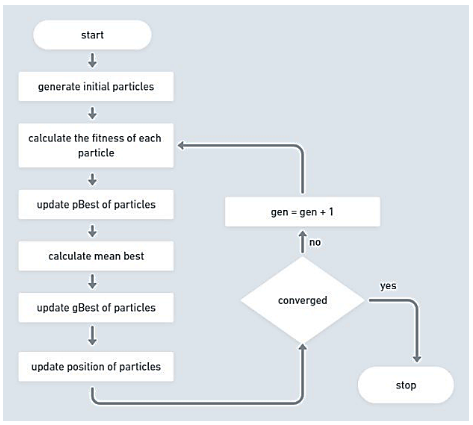
w: inertia weight

$xBest_i^t$: best particle position

$gBest_i^t$: best group position

III. METHODOLOGY

A flowchart showing the PSO algorithm:


start
generate initial particles
calculate the fitness of each particle
update pBest of particles
calculate mean best
update gBest of particles
update position of particles
gen = gen + 1
no
converged
yes
stop

As the beginning point for each particle's value, choose a random number from the possible choices. After that, we must calculate the fitness value for each individual particle. If the new fitness value is higher than the all-time best fitness value (pbest), then the new fitness value should be added to the new pbest. Choose, among all of the particles, the one that has the greatest fitness value to act as the best particle. Calculate the velocity of each particle by using the equation for updating the velocity, which states x=w*v+cl*rand()*(pbest-x)+c2*rand()*, to the data (gbest-x). After that, update the particle position by using the equation for position updates, which is x = x+v. The execution will be finished when there is a convergence in the position of the particles after they have been updated. Mostly in situation that the value has not yet converged, the execution will continue until it does.

IV. RESULT AND ANALYSIS

In this example of Particle Swarm Optimization, this method is comprised of fifty individual particles (PSO). The position of each particle in D-dimensional space represents a possible solution to the problem. There are three fundamental rules that control the state changes of each of these fifty particles.

(i) maintaining its state of inertia.

(ii) Altering the situation so that they correspond with the most optimistic state possible.

(iii) Changing the situation such that it is consistent with the swarm's most optimistic state on the future.

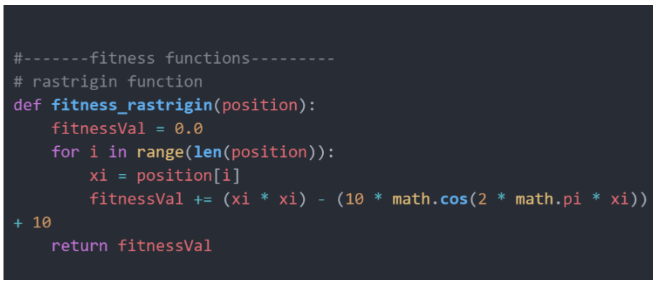
Variable	Symbol	Value	
		PSO	IPSO
Population Size	-	50	50
Maximum Iteration Size	-	100	100
Inertia Weight	w	0.704	0.704
Acceleration Constant	c1	1.457	1.457
	c2	1.543	1.543

Table 1. The Parameters of PSO and IPSO

The Rastrigin function is what's used to figure out whether or not a particle is healthy. By calculating the equation in the lines of code function below using the particle's position as an input, the fitness of a particle may be calculated.

$$f_7(x) = \sum_{i=1}^{n} \left[x_i^2 - 10\cos(2\pi x_i) + 10\right]$$

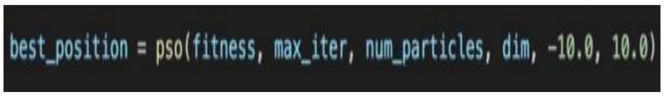
A point in the search space in the ***i*th** dimension is denoted by the sign $\mathbf{x_i}$, and each $\mathbf{x_i}$ refers to a different point. The value 0 is the optimal answer for this function. After each epoch, the fitness of a particle is re-evaluated by applying this function. If enough epochs occur, each particle will eventually settle into the position that corresponds to the function minimum.



```
#-------fitness functions---------
# rastrigin function
def fitness_rastrigin(position):
    fitnessVal = 0.0
    for i in range(len(position)):
        xi = position[i]
        fitnessVal += (xi * xi) - (10 * math.cos(2 * math.pi * xi))
+ 10
    return fitnessVal
```

Diagram 1. The Rastrigin Function

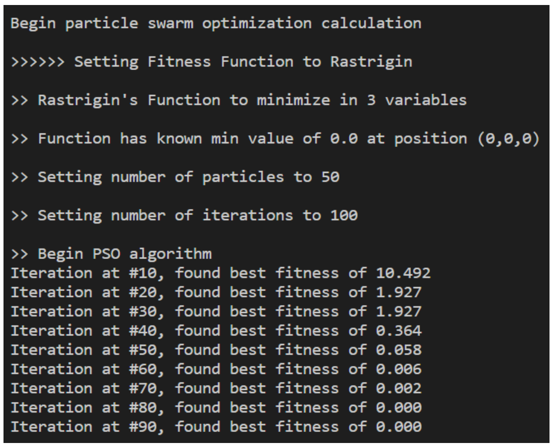
The Rastrigin's function in all three variables should be minimize as the purpose of this exercise. The function shown in Diagram 1 is the one that is used, and the line of code that determines the best position for this study is shown in Diagram 2, which may be seen below.



```
best_position = pso(fitness, max_iter, num_particles, dim, -10.0, 10.0)
```

Diagram 2. Best Position

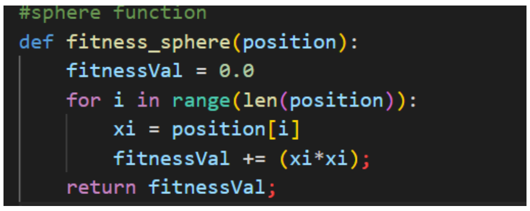
According to the PSO concept, the term of "best position" is only used with regard to the individual particle in issue over the process of the evolutionary process. This is done without taking into consideration the impact of fitness distribution across the landscape. The PSO will be carried out too early as a result of incorrect information on personal best rankings, which cannot be corrected. As a consequence of this, two new locations are developed, both of which increase the influence of optimal fitness distribution in order to improve a particle's ability to learn from the experiences of other particles. After that, a cognitive term consisting of three determined best locations is included into the equation for updating the velocity in order to reduce the chance of being misled. Diagram 3 shows the final outcome that we obtain, which is 0.0000 for the best possible fitness solution.



```
Begin particle swarm optimization calculation

>>>>>> Setting Fitness Function to Rastrigin

>> Rastrigin's Function to minimize in 3 variables

>> Function has known min value of 0.0 at position (0,0,0)

>> Setting number of particles to 50

>> Setting number of iterations to 100

>> Begin PSO algorithm
Iteration at #10, found best fitness of 10.492
Iteration at #20, found best fitness of 1.927
Iteration at #30, found best fitness of 1.927
Iteration at #40, found best fitness of 0.364
Iteration at #50, found best fitness of 0.058
Iteration at #60, found best fitness of 0.006
Iteration at #70, found best fitness of 0.002
Iteration at #80, found best fitness of 0.000
Iteration at #90, found best fitness of 0.000
```

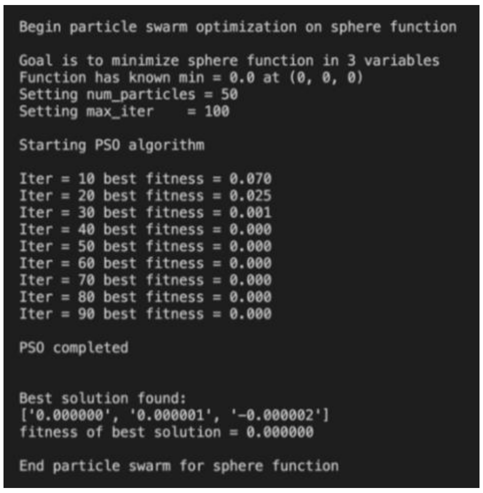
DIAGRAM 3. RESULT OF RASTRIGIN'S FUNCTION

The Rastrigin function and the sphere function are commonly used benchmark functions for optimization algorithms. There are a few different performance metrics that can be used to evaluate the performance of a Particle Swarm Optimization (PSO) algorithm on these functions. One metric is the number of function evaluations required to find a solution that meets a certain accuracy threshold. This measures the efficiency of the algorithm in terms of the number of function evaluations required to find a satisfactory solution. Another metric is the final value of the objective function. For the Rastrigin function, the optimal value is 0, so the closer the final value is to 0, the better the algorithm has performed. For the sphere function, the optimal value is also 0, so again, the closer the final value is to 0, the better the algorithm has performed. Other metrics that can be used to evaluate the performance of a PSO algorithm on the Rastrigin function and the sphere function include the convergence speed of the algorithm, the robustness of the algorithm (i.e., how well it performs on a range of different initializations and problem configurations), and the quality of the final solution found by the algorithm.



```
#sphere function
def fitness_sphere(position):
    fitnessVal = 0.0
    for i in range(len(position)):
        xi = position[i]
        fitnessVal += (xi*xi);
    return fitnessVal;
```

Diagram 4. Sphere Function



```
Begin particle swarm optimization on sphere function

Goal is to minimize sphere function in 3 variables
Function has known min = 0.0 at (0, 0, 0)
Setting num_particles = 50
Setting max_iter     = 100

Starting PSO algorithm

Iter = 10 best fitness = 0.070
Iter = 20 best fitness = 0.025
Iter = 30 best fitness = 0.001
Iter = 40 best fitness = 0.000
Iter = 50 best fitness = 0.000
Iter = 60 best fitness = 0.000
Iter = 70 best fitness = 0.000
Iter = 80 best fitness = 0.000
Iter = 90 best fitness = 0.000

PSO completed


Best solution found:
['0.000000', '0.000001', '-0.000002']
fitness of best solution = 0.000000

End particle swarm for sphere function
```

Diagram 5. Result of Sphere Function

The Sphere Function lines of code that were used for this research are shown in Diagram 4. The result for the Sphere Function can be seen in Diagram 5, and it was achieved by reusing the Best Position line of code from Diagram 2.

V. Discussion and Conclusion

Particle Swarm Optimization, also known as PSO, is a method that is used widely in a variety of industries, including image processing, engineering design, and artificial intelligence, among others. The main goal of optimization is to determine the values of the variables that minimize the objective function, also known as "optimal" designs, while still satisfying the constraints. To put it another way, the goal of optimization is to select the most effective option from among a set of possible solutions. If we compare continuous optimization with discrete optimization, we find that discrete optimization problems are more challenging to resolve than continuous optimization challenges. Utilizing individual best practices has several aims, one of which is to increase the range of outstanding solutions. Randomization is one method that may be used to simulate this diversity. Unless the optimization issue is significantly nonlinear and multimodal, there is no strong reason to choose a single optimum option. The phrase "climate change" refers to the periodic changes in temperature and pattern that have been seen over long periods of time. It's possible that these shifts are the result of natural forces, including changes in the solar cycle. The carbon dioxide emissions, such as carbon dioxide and methane, are two examples of the factors that lead to climate change. As a direct result of this, both the consumption of energy and the output of greenhouse gases will increase. It is important to do research on the most effective layout for a heating system in order to bring down the overall cost of the project and the amount of energy required for heating. The main purpose of this study is to explore the possible applications of the PSO algorithm in the process of reducing the cost of heating systems, which is the issue that has to be addressed when developing heating systems. The results demonstrate that PSO is an useful tool for optimizing planning procedures for heating systems.

In conclusion, we are capable of designing and simulate python coding that is based on particle swarm optimization that minimizes the rastrigin and sphere functions. The Rastrigin function is what's used to figure out whether or not a particle is healthy. The PSO will be carried out too early as a result of incorrect information on personal best positions, which cannot be corrected. Since there is only one objective function that needs to be taken into consideration, the sphere function is perfect for single-objective optimization.

VI. REFERENCE

[1] United Nations. (n.d.). *What Is Climate Change?* https://www.un.org/en/climatechange/what-is- climate-change

[2] Wang, Y. F., Li, K. P., Xu, X. M., & Zhang, Y. R. (2014). Transport energy consumption and saving in China. *Renewable and Sustainable Energy Reviews, 29*(December 2017), 641–655. https://doi.org/10.1016/j.rser.2013.08.104

[3] Yang, X.-S., Fong Chien, S., & Ting, T. O. (2014). *Editorial Computational Intelligence and Metaheuristic Algorithms with Applications*. https://doi.org/10.1155/2014/425853

[4] Cheng, S., Lu, · Hui, Lei, · Xiujuan, Shi, · Yuhui, Lu, B. H., Lei, X., &#38; Shi,Y. (2018). A quarter century of particle swarm Intelligent Systems 227–239. https://doi.org/10.1007/s40747-018-0071-2</div

[5] *What is Optimization | IGI Global*. (n.d.). IGI Global. https://www.igi-global.com/dictionary/cuckoo-search-for-optimization-and-computational-intelligence/21383

[6] Kelley, T. R. (2010). *Optimization, an Important Stage of Engineering Design*. https://digitalcommons.usu.edu/ncete_publications

[7] *Types of Optimization Problems & Technique |Prescient Technologies*. (n.d.). Retrieved January 8, 2022, from https://www.pre-scient.com/knowledge-center/optimization-problems/optimization-problems.html

[8] *Particle Swarm Optimization -an overview | ScienceDirect Topics*. (n.d.). Retrieved January 8, 2022, from https://www.sciencedirect.com/topics/engineering/particle-swarm-optimization

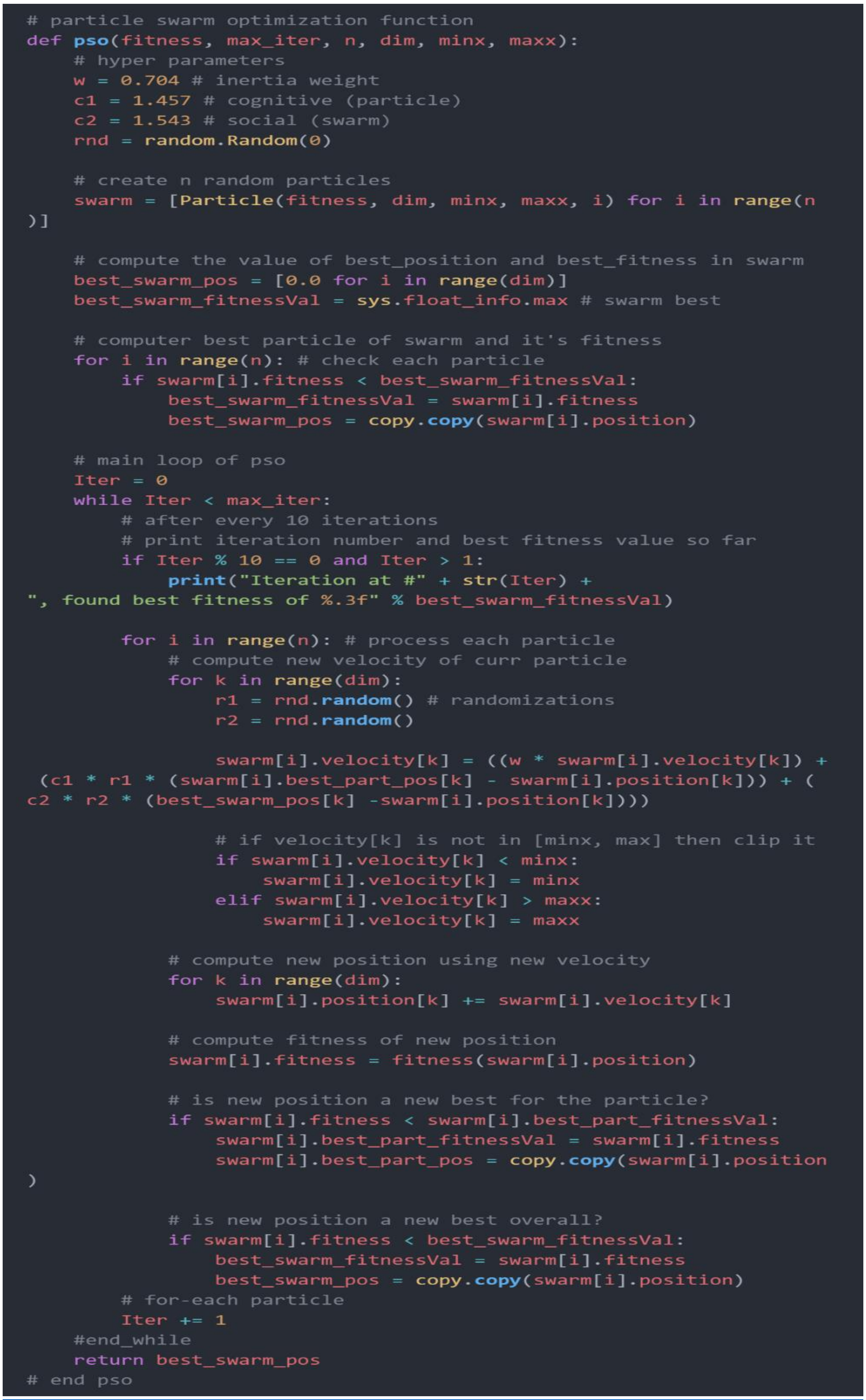
[9] S. (n.d.). *GitHub - stratzilla/pso-rastrigin: Particle swarm optimization of Rastrigin function*. GitHub. Retrieved January 8, 2022, from https://github.com/stratzilla/pso-rastrigin

[10] GeeksforGeeks.(2021, August 31). *Implementation of Particle Swarm Optimization*.Retrieved January 8, 2022, from https://www.geeksforgeeks.org/implementation-of-particle-swarm-optimization/

[11] User, S. (n.d.). Sphere function (pure excel). xloptimizer.com. Retrieved January 8, 2022, from https://xloptimizer.com/projects/toy-problems/sphere-function-pure-excel

[12] *A Gentle Introduction to Particle Swarm Optimization*. (n.d.). Retrieved January 8, 2022, from https://machinelearningmastery.com/a-gentle-introduction-to-particle-swarm-optimization/

CODING



```
# particle swarm optimization function
def pso(fitness, max_iter, n, dim, minx, maxx):
    # hyper parameters
    w = 0.704 # inertia weight
    c1 = 1.457 # cognitive (particle)
    c2 = 1.543 # social (swarm)
    rnd = random.Random(0)

    # create n random particles
    swarm = [Particle(fitness, dim, minx, maxx, i) for i in range(n
)]

    # compute the value of best_position and best_fitness in swarm
    best_swarm_pos = [0.0 for i in range(dim)]
    best_swarm_fitnessVal = sys.float_info.max # swarm best

    # computer best particle of swarm and it's fitness
    for i in range(n): # check each particle
        if swarm[i].fitness < best_swarm_fitnessVal:
            best_swarm_fitnessVal = swarm[i].fitness
            best_swarm_pos = copy.copy(swarm[i].position)

    # main loop of pso
    Iter = 0
    while Iter < max_iter:
        # after every 10 iterations
        # print iteration number and best fitness value so far
        if Iter % 10 == 0 and Iter > 1:
            print("Iteration at #" + str(Iter) +
", found best fitness of %.3f" % best_swarm_fitnessVal)

        for i in range(n): # process each particle
            # compute new velocity of curr particle
            for k in range(dim):
                r1 = rnd.random() # randomizations
                r2 = rnd.random()

                swarm[i].velocity[k] = ((w * swarm[i].velocity[k]) +
 (c1 * r1 * (swarm[i].best_part_pos[k] - swarm[i].position[k])) + (
c2 * r2 * (best_swarm_pos[k] -swarm[i].position[k])))

                # if velocity[k] is not in [minx, max] then clip it
                if swarm[i].velocity[k] < minx:
                    swarm[i].velocity[k] = minx
                elif swarm[i].velocity[k] > maxx:
                    swarm[i].velocity[k] = maxx

            # compute new position using new velocity
            for k in range(dim):
                swarm[i].position[k] += swarm[i].velocity[k]

            # compute fitness of new position
            swarm[i].fitness = fitness(swarm[i].position)

            # is new position a new best for the particle?
            if swarm[i].fitness < swarm[i].best_part_fitnessVal:
                swarm[i].best_part_fitnessVal = swarm[i].fitness
                swarm[i].best_part_pos = copy.copy(swarm[i].position
)

            # is new position a new best overall?
            if swarm[i].fitness < best_swarm_fitnessVal:
                best_swarm_fitnessVal = swarm[i].fitness
                best_swarm_pos = copy.copy(swarm[i].position)
        # for-each particle
        Iter += 1
    #end_while
    return best_swarm_pos
# end pso
```