1. PARTICLE SWARM OPTIMIZATION

Particle swarm optimization (PSO) was developed by Kennedy and Eberhart in 1995 based on swarms founded in nature, such as bird flock and fish shoal. Since then, PSO has generated much wider interests and forms an exciting, ever-expanding research subject, called swarm intelligence. PSO has been applied to almost every area in optimization, computational intelligence, and design applications. There are at least two dozen PSO variants, and hybrid algorithms by combining PSO with other existing algorithms are also investigated extensively [1].

* 1. Swarm intelligence

The Swarm intelligence ​is an artificial intelligence (AI) field which studies the behavior of decentralized and self-organized systems, understanding by self-organization the ability of some physical systems formed by many individuals, to create behavior patterns adaptable and not predictable, without a central intelligence. Observing social organizations found in nature, as well as the behavioral characteristics of their colonies, inspires many swarm intelligence algorithms such as Ant Colony Optimization (ACO) and Firefly Optimization Algorithm (FOA), e.g.

The PSO become one of the most popular swarm-intelligence-based algorithms due to its programming simplicity and flexibility face of several problems. Instead of using mutation and crossover, techniques applied in Genetic Algorithms (GA), PSO uses real-numbers randomness and collective communication between particles. Thus, there is no decoding of the parameters in binary numbers such as those present in AG.

Many new algorithms that are based on swarm intelligence may have drawn inspiration from different sources, but they have some similarity to some of the components that are used in PSO. In this sense, PSO pioneered the basic ideas of swarm-intelligence based computation [2].

* 1. PSO algorithm

The PSO algorithm consists in particle sets with individuals, where each is random positioned within a parameters space (limiting a feasible region), each representing a possible solution of proposed problem. Each particle has velocity which changes its position through the feasible region using a series of discrete timesteps (iterations), in this work the PSO main algorithm was implemented in MATLAB. The velocity vector for each particle is adjusted at each timestep according to the best individual performance of that specific particle, as well as best performance of the swarm as a whole. The performance of each new candidate solution is quantified using an objective function, and the process is repeated until the convergence criteria are met. In this work, was adopt a swarm size and a maximum iteration .

The potential solutions are represented by a vector , which comprises a specific parameter set entering the possible solutions of a problem within a feasible region. In the PSO, the vector represents a particle position. The particular particle position at each timestep is given by and the velocity of particle at each timestep is given by . The objective function determines the performance of the position of each particle, the best value of objective function of any particle in the swarm at time is given by ,the historical best value of objective function in the swarm particle until the time is given by . The global best position of any particle in the swarm at time is given by , while the historical best position in the swarm particle until the time is given by .

The velocity of each particle is updated at each timestep according to Eq. (16), following:

(16)

where and are random numbers from a Gaussian distribution with values between 0 to 1, and are cognitive acceleration factor and the social acceleration factor, respectively, and is the inertia factor. These weighting factors are bounded according to the rules of Eq. (17) for algorithm convergence.

0

.

The inertia term , determines how much of the speed from the previous timestep is over to the next one. High inertia values ​​cause particles to behave more independently and explore the solution space more meticulously, while lower values ​​cause faster swarm convergence. One popular strategy for selecting the inertia term is to use a dynamic factor that begins at high value, and gradually decreases during algorithm interactions. This approach brings with it the advantage of a timely convergence while to force the particles to fully explore the solution space [3].

The cognitive acceleration factor (also known as the cognitive factor), multiplied by a random number , determines the influence of the best historical performance of each individual particle. The social acceleration factor (also known as the swarm factor) multiplied by a random number , determines the influence of the best historical performance of swarm. A high social acceleration factor value causes a quicker algorithm convergence to the best swarm position, but limits individual particle exploration. On the other hand, a high self-confidence parameter causes each particle to fully explore any optimal regions each particle encounters, but may delay convergence. It is common for the cognitive acceleration factor to be equal to or slightly larger than the swarm influence parameter (in the range of 1.5 to 2.5), due to the convergence criteria in Eq. (17) leads to the best behavior of the PSO algorithm.

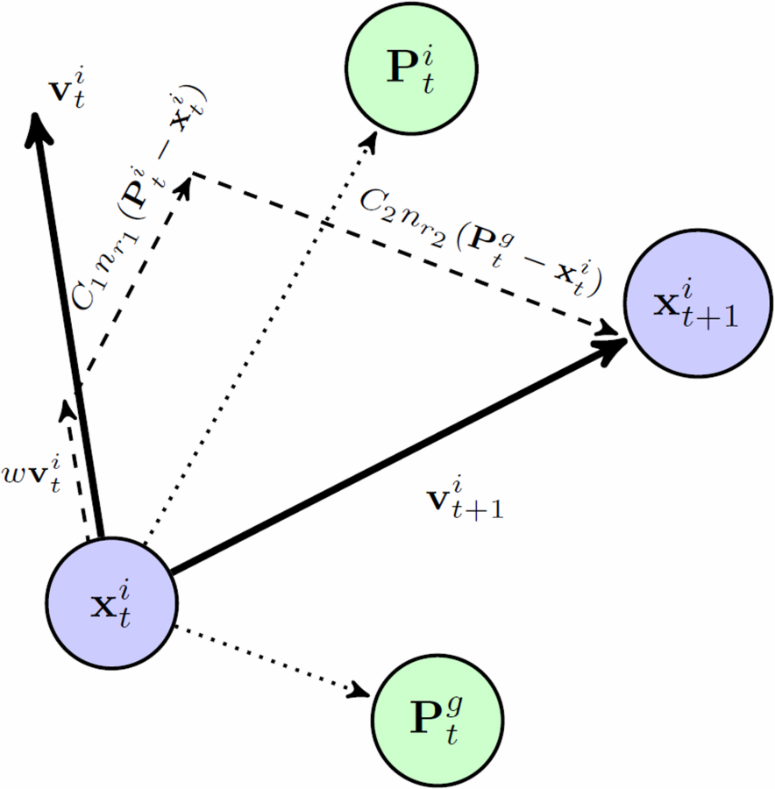


Figure 1: Particle’s movement in the PSO

Figure 1 schematically illustrates particle’s movement in the PSO algorithm. The essence steps of optimization algorithm through particle examination can be summarized in the pseudocode shown in Fig. 2.

**PARTICLE SARW OPTIMIZATION**

1: Objective function  **,**

2: Initialize locations and velocity of particles

3: Finding from min at

4: ***while*** (criterion)

5: ***for***loop over all particles and **all** dimensions

6: Calculate new velocity

7: Calculate new locations

8: Evaluate objective functions at new locations

9: ***end for***

10: Find the current global best

11: Update (pseudo time or iteration counter)

11: ***end while***

12: Output the final results  and

Figure 2: PSO pseudocode Figure [4]

* 1. Accelerated PSO (APSO)

The standard PSO uses both best the current global best and the individual best. Using the individual best mostly to increase the diversity in the quality solutions. But this diversity can be simulated using some randomness. Thus, there is no compelling reason for using the individual best unless the optimization problem of interest is highly nonlinear and multimodal [4].

A simplified PSO version which uses only the global best to accelerates algorithm’s convergence, is called of Accelerated Particle Swarm Optimization(APSO), its velocity vector is given by:

(18)

where is a random variable with values from 0 to 1. Here the shift 1/2 is purely out  
of convenience. We can also use a standard normal distribution , where is drawn  
from N (0, 1) to replace the second term. Now we have

(19)

where can be drawn from a Gaussian distribution or any other suitable distributions.  
The update of the position is simply

. (20)

We can also simplify the formulation writing the update of the location in a single step

. (21)

The typical initial values for the PSO are and , for unimodal objective functions. It is worth pointing out that the parameters and should in general be related to the scales of the independent variables and the search domain.

In addition, the APSO brings an improvement reducing the randomness as interaction process proceeds. Usually using a monotonically decreasing function such as:

(22)

where is the initial value of the randomness parameter. Here is the timesteps number, is a control parameter. In this work, was adopt .

* 1. Convergence analysis

From the statistical point of view, each particle in PSO forms a Markov chain, though this Markov chain is biased toward the current best, since the transition probability often leads to the acceptance of the move toward the current global best. In addition, the multiple Markov chains are interacting in terms of partially deterministic attraction movement. Therefore, any mathematical analysis concerning the rate of convergence of PSO may be difficult. However, there are some good results using both the dynamical system and Markov chain theories.

* + 1. Dynamical system

The first convergence analysis in terms of dynamical system theories was carried out by Clerc and Kennedy in 2002 [2]. Mathematically, if we ignore the random factors, we can view the system formed by (7.1) and (7.2) as a dynamical system. If we focus on a single particle and imagine there is only one particle in this system, then the global best is the same as its current best . In this case, we have

and

Following the analysis of a 1D dynamical system for particle swarm optimization, we can replace with a parameter constant p so that we can see whether or not the particle of interest will converge toward . Now we can write this system as a simple dynamical system.

For simplicity, we focus on only a single particle. By setting and using the notations for dynamical systems, we have

or

where

, ,

The general solution of this dynamical system can be written as

The main behavior of this system can be characterized by the eigenvalues of

It can be seen clearly that leads to a bifurcation.

Following a straightforward analysis of this dynamical system, we can have three cases. For , cyclic and/or quasi-cyclic trajectories exist. In this case, when randomness is gradually reduced, some convergence can be observed. For , noncyclic behavior can be expected, and the distance from to the center (0, 0) is monotonically increasing with t. In a special case , some convergence behavior can be observed. Since p is linked with the global best, as the iterations continue it can be expected that all particles will aggregate toward the global best.

However, this global best is the best solution found by the algorithm during iterations,  
which may be different from the true global optimality of the problem of interest. This  
point will also become clearer in the framework of Markov chain theory.

* + 1. Markov chain approach

Various studies on the convergence properties of the PSO algorithm have diverse results [2,9–12]. However, care should be taken in interpreting these results in practice, especially for the discussion of the implications from a practical perspective. Many studies can prove the convergence of the PSO under appropriate conditions, but the converged states are often linked to the current global best solution g∗ found so far. There is no guarantee that this is the true global solution true to the problem. In fact, many simulations and empirical observations by many researchers suggest that in many cases, this g∗ is often stuck in a local optimum, which is often referred to as the premature convergence. All these theories do not guarantee

In fact, many statistically significant test cases suggest that

The standard PSO does not guarantee that the global optimum is reachable, and the global optimality is searchable with a certain probability ), where is the th iteration and is the swarm size. Here is the state sequences of the particles.

For a given swarm size m and iteration k, a Markov chain formed by the th particle at time can be defined, and then the swarm state sequence also forms a Markov chain for all . For the standard PSO with an inertia parameter shown in Eq. (7.3), the probability that is in the optimal set can be estimated by

where is a local radius and is defined by *,*. Pan et al. showed  
that

which means that the standard PSO will not guarantee to get to the global optimality nor guarantee to miss it. The equality only holds for and . Since the size is finite and the iteration is also finite, this probability is typically . However, that the standard PSO does not guarantee convergence does not mean that other variants cannot have global convergence. In fact, a recent study used the framework of Markov chain theory and proved that APSO can have global convergence [5]

* 1. Selected numerical studies using PSO

The range of modifications of the basic PSO method we have described clearly need to be examined to see how effective they are. To evaluate PSO performance, three benchmark problems illustrating specific features of optimization. The first problem, Rosenbrock’s function which has only one local minimum, the second problem due to Styblinski and Tang has a small number of local optima and a specific global minimum the third, Rastrigin’s function has many local minima and consequently presents a harder test for finding the global minima. None of these problems demand excessive computer time for low dimensional problems and consequently may be easily replicated by the reader.

Rosenbrock’s function is a classic test problem for gradient optimization methods and because the single optimum is located in a long shallow curved valley, it presents significant difficulties and takes many iterations to locate the optimum. It is used in these tests to show that the methods can also solve difficult single optima problems. The minimum value is and is obtained when for . The problem is usually defined for test purposes in the range [−5, 5] for each variable, although the range [−5, 10] is sometimes used.

The Styblinski-Tang function for two variables is not a demanding problem but has various local minima so it is a useful test to see that the global minimum is found at the point  and . With global minimum at 78.3323. (See Figure 3.6.) The problem is usually tested in the range of values for the independent variables and in the range [−5, 5].

The final problem we can consider in this section is Rastrigin’s function which is a more demanding problem since it has many closely packed local minima of similar value and a significant danger of an optimization method becoming trapped in a particular local minimum rather than finding the global minimum. This has a global minimum at x = [0, 0]T with . The problem is usually tested in the range [−5, 5] for the independent variables.

We now use a modified version of the PSO algorithm and test it on the functions we

have described in various ways. Figure 3.1 shows the convergence for the much more

difficult problem of minimizing the Rastrigin function in 6 dimensions or variables. It shows a gradual reduction in the function value over the 2000 iterations and appears to indicate periods where little change occurs in the current best value of the objective function as the area is searched for improvements.

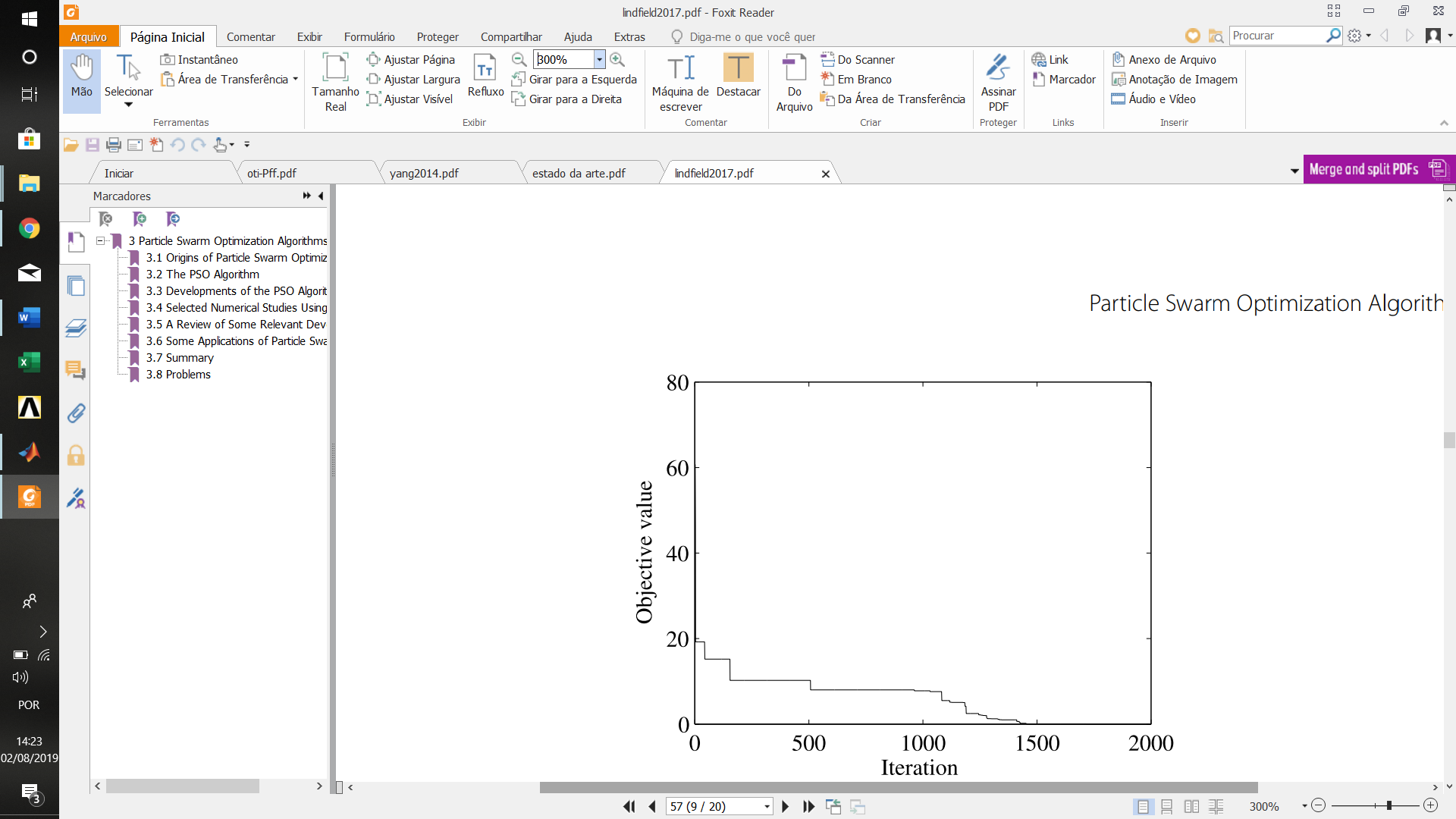


Figure 3: Minimizing Rastrigin’s function with 6 variables, showing progress of convergence.

Table 1: Effect of number of iterations on the estimate of the minimum value of the function. C1= 2.05 and C2 = 2.05

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Iteration | Rosenbrock | Styblinski | Rastrigin  (2D) | Rastrigin  (4D) |
| 200 | 3.36E-04 | -7.83E+01 | 1.13E-09 | 2.21E+00 |
| 400 | 4.46E-07 | -7.83E+01 | 0.00E+00 | 2.05E-02 |
|  |  |  |  |  |

Table 2: Effect of swarm size on the estimate of the minimum value of the function after 200 iterations. C1= 2.05 and C2 = 2.05

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Swarm  Size | Rosenbrock | Styblinski | Rastrigin  (2D) | Rastrigin  (4D) |
| 20 | 3.36E-04 | -7.83E+01 | 1.13E-09 | 2.21E+00 |
| 10 | 6.34E-04 | -7.83E+01 | 5.76E-07 | 2.28E+00 |

The basic features that effect the performance of the algorithm are the number of iterations and the size of the swarm. In Table 3.1 we give some results for the test problems that show the difference between using 200 iterations and 400 iterations. In Tables 3.1 and 3.2, ROS2 indicates Rosenbrock’s function with two variables, S-T2 indicates the Styblinski-Tang function in two variables and RAS2 and RAS4 indicates Rastrigin’s function in two and four variables, respectively.

Recalling that the solution of the Rosenbrock and the two and four variable Rastrigin functions are zero and the solution of the Styblinski and Tang is −78.3323, Table 3.1 shows small or no improvements in the minimum values except for the last problem which is a more demanding one with four variables and shows a very large improvement with 400 iterations.

Similarly the effect of swarm size can be studied. Various researchers have recommended different swarm sizes, usually 10, 20 or 30 but sometimes more, depending on the problem. Swarm size is very important parameter since too few members of the swarm give poor results rapidly but too many members generally give more accurate results more slowly. Indeed a very large swarm size amounts to little more than enumeration of the possible solutions and a complete loss of efficiency.

Table 3: Minimization of the Rastrigin’s function (4D) . 800 iterations

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Swarm Size | Mean | Best | Worse | St Dev |
| 10 | 5.97E-01 | 0.00E+00 | 1.99E+00 | 7.50E-01 |
| 20 | 3.45E-01 | 0.00E+00 | 5.90E+00 | 1.33E+00 |
| 30 | 1.07E-15 | 0.00E+00 | 2.14E+14 | 4.77E-13 |

Table 4: Minimization of the Rastrigin’s function (4D) .2000 iterations

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Swarm Size | Mean | Best | Worse | St Dev |
| 10 | 4.97E-02 | 0.00E+00 | 9.95E-01 | 2.25E-02 |
| 20 | 2.57E-12 | 0.00E+00 | 5.15E-11 | 1.15E-11 |
| 30 | 0.00E+00 | 0.00E+00 | 0.00E+00 | 0.00E+00 |

Table 5: Minimization of the Rastrigin’s function (6D) .2000 iterations

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Swarm Size | Mean | Best | Worse | St Dev |
| 10 | 1.33E+00 | 0.00E+00 | 3.75E+04 | 1.06E+00 |
| 20 | 5.47E-01 | 0.00E+00 | 1.99E+00 | 7.55E-01 |
| 30 | 2.00E-01 | 0.00E+00 | 9.95E-01 | 4.08E-01 |

Table 3.2 sets the number of iterations at 200 and illustrates the effect of different  
swarm sizes. The table shows only slightly less accurate results for the smaller swarm size for those problems which are not demanding. A more thorough test is to perform many runs of the algorithm and take the mean of the results, thus smoothing out some of the effects of randomness. The results shown in Table 3.3 and 3.4 for the minimization of the RAS4 test function, using 20 runs. Two tests are performed, one with 800 iterations and one with 2000 iterations. Since the  
minimum value of this function is zero, it is clear that the larger number of iterations  
improves the accuracy of the solution and the larger the swarm size the better the  
optimum obtained for this particular function. To provide further information about  
the performance of the method we have included the best and worst results, together  
with the mean and the standard deviation of these results.

As a further illustration we minimize the Rastrigin function but with six variables  
(RAS6), a much harder problem. In this set of runs we have used 2000 iterations and  
swarm sizes 10, 20 and 30. Taking 20 runs gives the results shown in Table 3.5. Contrast  
these results with those of Table 3.4 which illustrates how the algorithm performs on  
the same function but with only four variables rather than six. The results shown in  
Table 3.4 are clearly much better and illustrate how increasing the number of variables in a problem disproportionately increases the problems difficulty. This is frequently called the “curse of dimensionality”.

Table 6: Effect of parameters *C1* and *C2* on optimization of Rastrigin’s function (4D) .800 iterations

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Set | Mean | Best | Worse | St Dev |
| 1 | 5.67E-02 | 0.00E+00 | 9.95E-01 | 2.23E-01 |
| 2 | 2.99E-01 | 0.00E+00 | 9.95E-01 | 4.68E-01 |

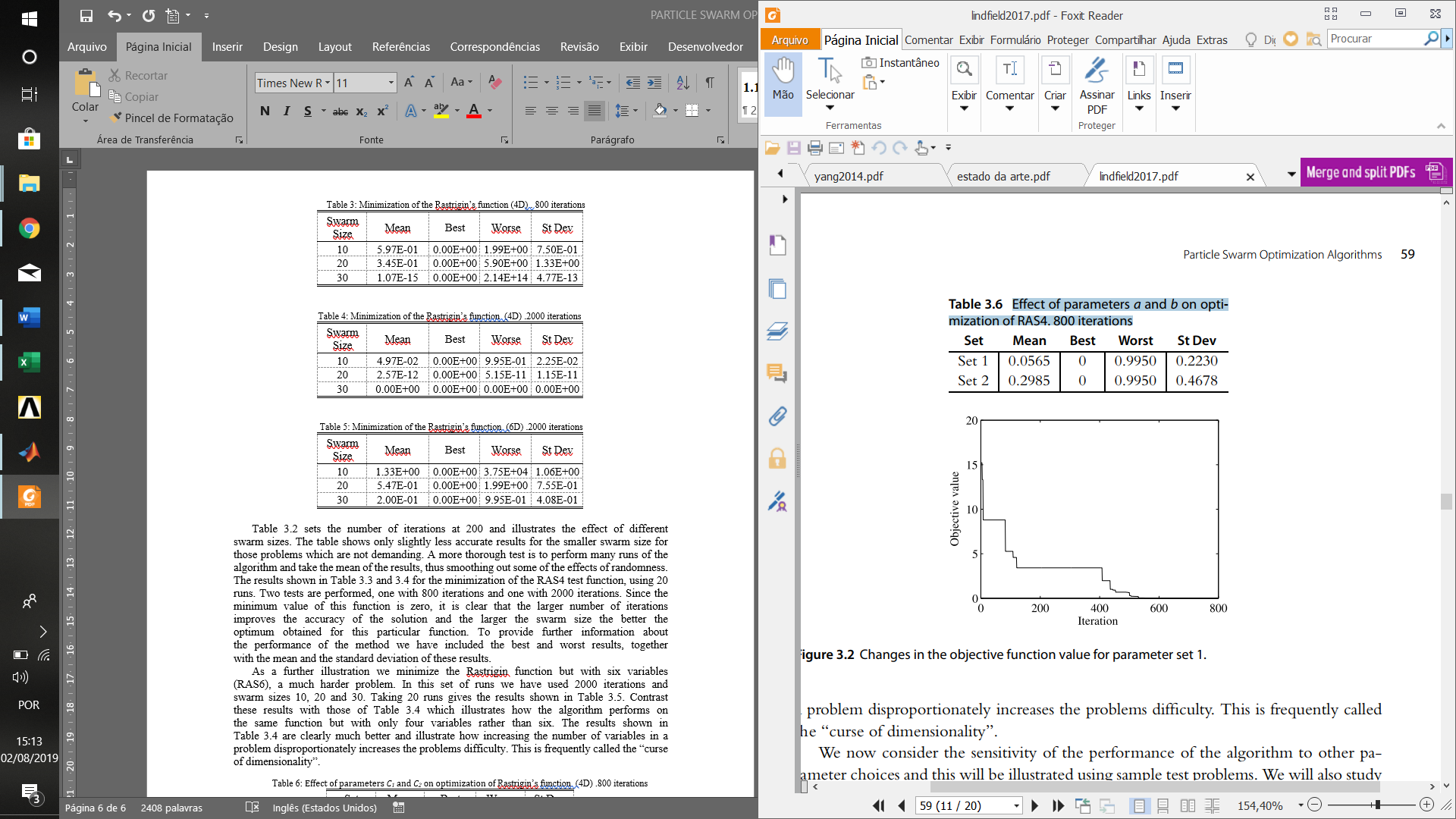


Figure 4: Changes in the objective function value for parameter set 1

We now consider the sensitivity of the performance of the algorithm to other parameter choices and this will be illustrated using sample test problems. We will also study the effects of some of the modifications to the algorithm suggested by workers in the field. This will be achieved by considering the effects of these modifications individually on the behavior of the algorithm and comparing the results for some test problems. Specifically we will consider the importance of the choice of the parameters a and b on the performance of the algorithm and the alternatives methods for setting the values of the inertial weights, w. The first study compares the performance of the PSO algorithm using values of the parameters a = 2.05 and b = 2.05, called set 1, with the use of a = 0.7 and b = 1.4, called set 2. A swarm size of 30 is used with 800 iterations. The function considered is the Rastrigin function with 4 variables, denoted by RAS4. The results for 20 runs is given in Table 3.6. There is significant improvement with set 1.

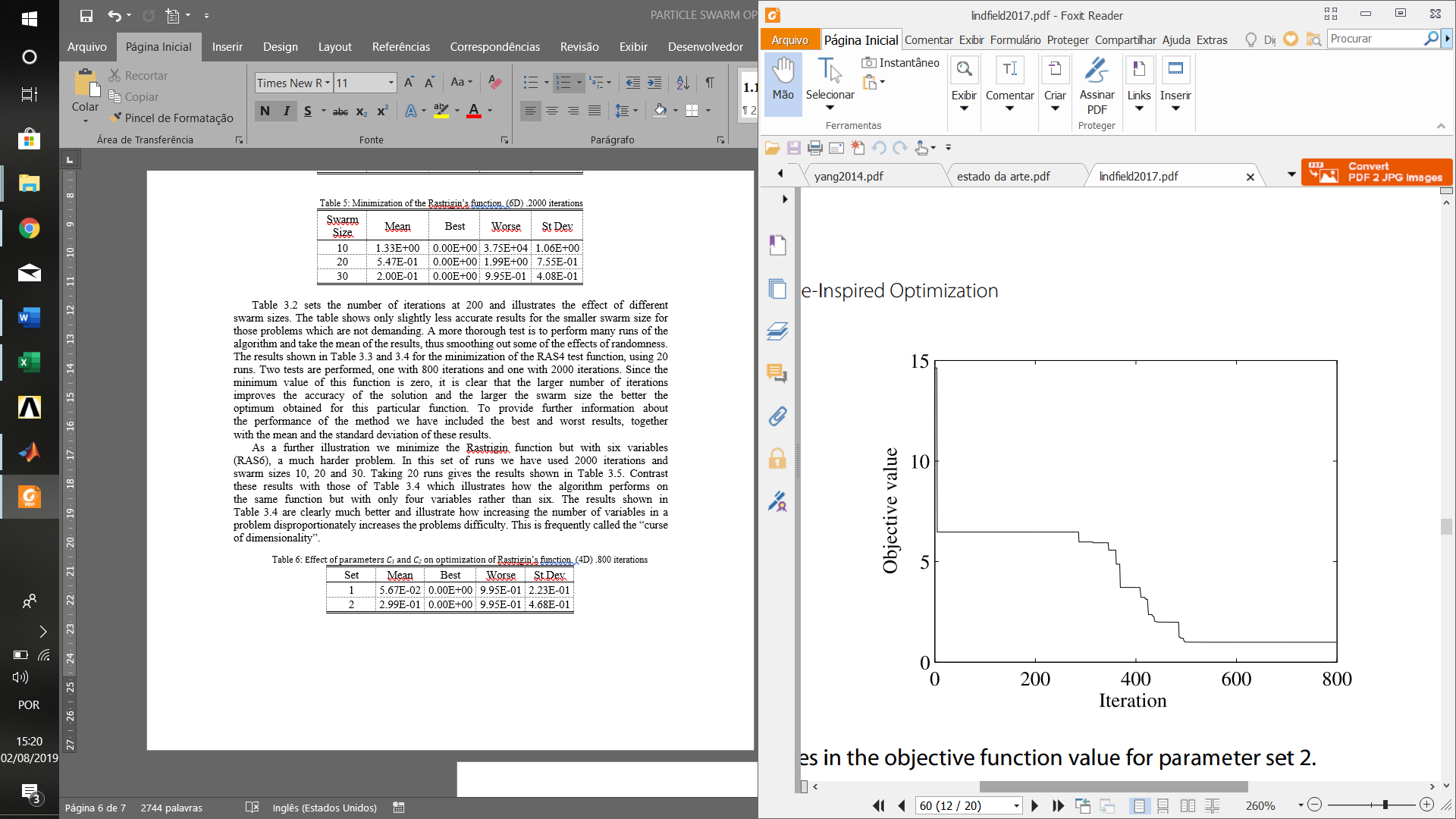


Figure 5: Changes in the objective function value for parameter set 2

As a graphical illustration, Figure 3.2 and Figure 3.3 show the progress of the iterations for an individual run for the two parameter sets. There is significant difference between the convergence paths but this could be because of the random nature of the process since it only involves one run of the PSO algorithm.

A further test considers how taking a specific value for the inertial weight compares with the continuous adjustment of the weight as the procedure continues. A swarm size of 30 is used with 800 iterations. The function considered is the Rastrigin function with 4 variables, RAS4. The results for 20 runs of the algorithm are given in Table 3.7. The table shows that there does seem to be a significant difference between the results, although further extensive testing should be performed before drawing hard and fast conclusions.

Figure 3.4 illustrates the progress of algorithm for the RAS4 function using w fixed at 0.7. Figure 3.5 shows the progress using w varied as the process proceeds

.

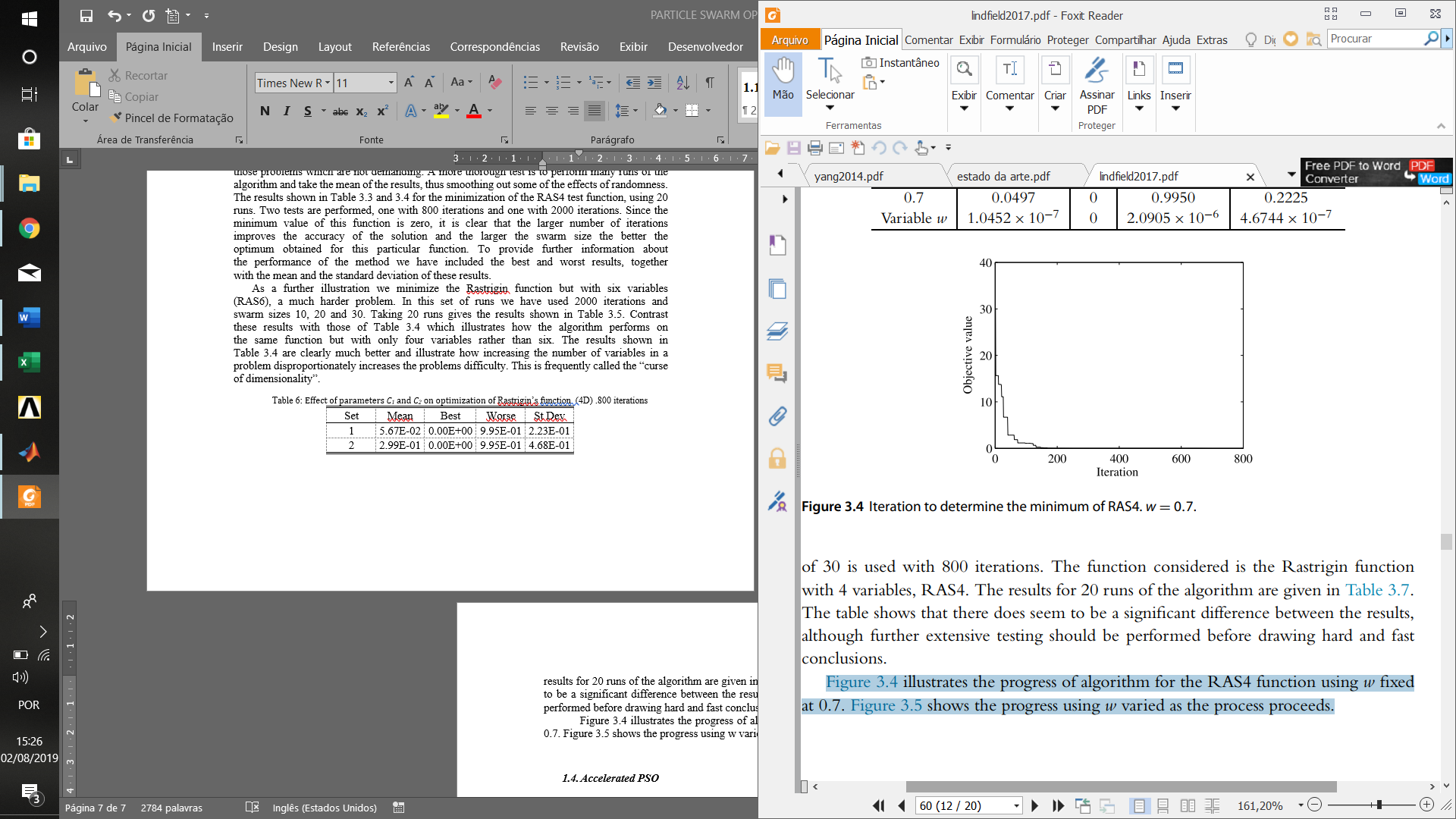


Figure 6: Iteration to determine the minimum of RAS4. w = 0.7

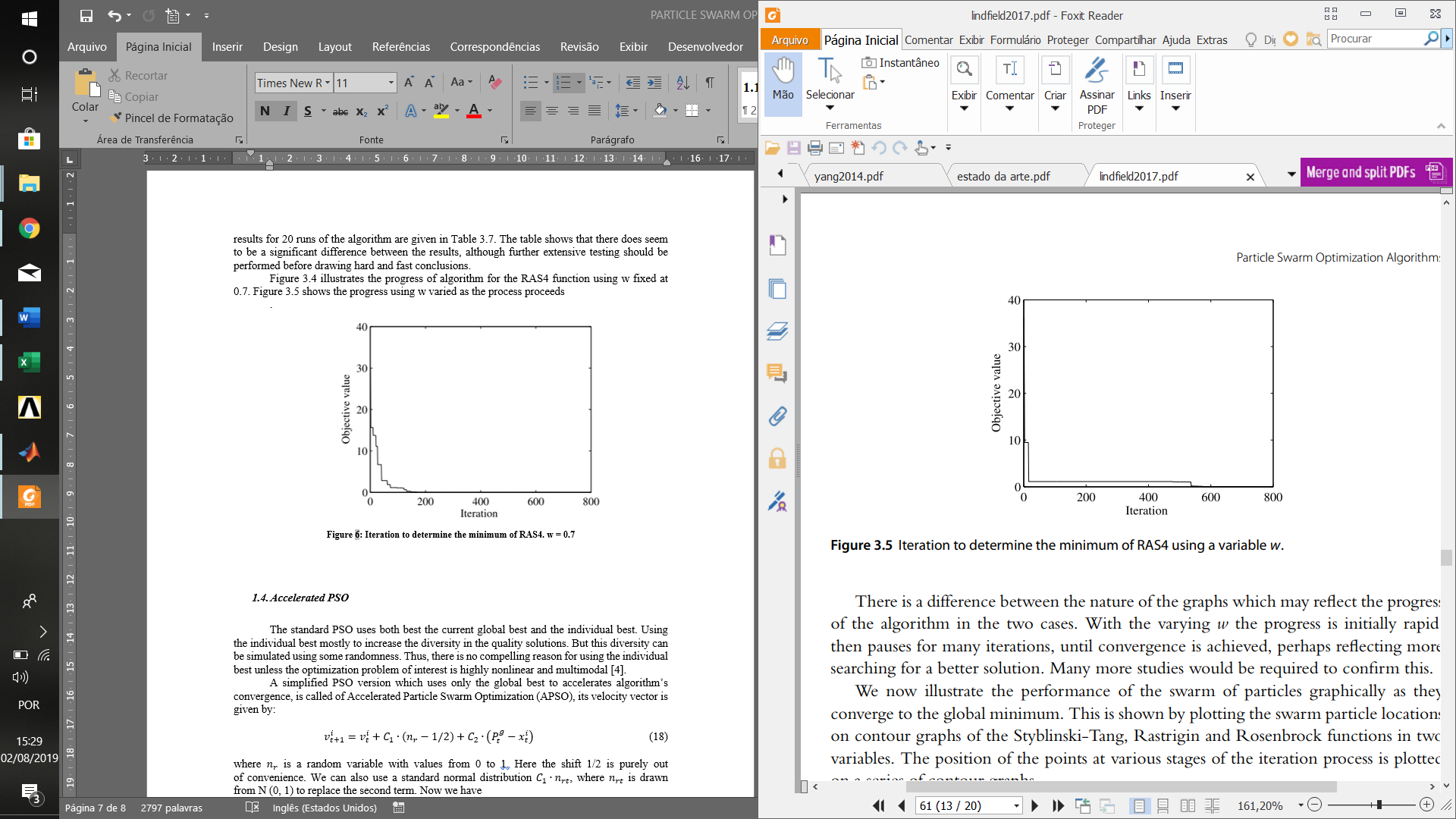


Figure 7: Iteration to determine the minimum of RAS4 using a variable w

There is a difference between the nature of the graphs which may reflect the progress  
of the algorithm in the two cases. With the varying w the progress is initially rapid, then pauses for many iterations, until convergence is achieved, perhaps reflecting more  
searching for a better solution. Many more studies would be required to confirm this. We now illustrate the performance of the swarm of particles graphically as they converge to the global minimum. This is shown by plotting the swarm particle locations on contour graphs of the Styblinski-Tang, Rastrigin and Rosenbrock functions in two variables. The position of the points at various stages of the iteration process is plotted on a series of contour graphs.

Table 7: Effect of constant and variable weight, w in finding the minimum of RAS4

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Set | Mean | Best | Worse | St Dev |
| 0.7 | 4.97E-02 | 0.00E+00 | 9.95E-01 | 2.23E-01 |
| Variable *w* | 1.05E-07 | 0.00E+00 | 2.09E-06 | 4.67E-07 |

We begin with the Styblinski-Tang function by plotting the initial particle position  
values, then the position of the particles after fifty, one hundred and three hundred  
iterations, each set of particles on a separate contour plot of the function. This is shown  
in Figure 3.6. We note that these graphs show rapid convergence to the global minimum of the  
function at the point (-2.9035, -2.9035). In Figure 3.6, 3.7 and 3.8 the individual particles are shown by filled circles.

A second example illustrates the convergence behavior for the Rastrigin function  
which has many closely packed minima in the region x = -5 to 5, y = -5 to 5. Figure 3.7 shows the convergence of the swarm. For clarity only the locations of the many  
minima are shown, rather than a conventional contour plot. Effective convergence to  
the global minimum at (0, 0) is achieved after three hundred iterations and convergence  
to the many nearby local minima is avoided.

As a final example we show how the algorithm behaves with Rosenbrock’s function which has a single unique minimum at (1, 1). The results for a run with the PSO algorithm for the optimization of Rosenbrock’s function are shown in Figure 3.8. The contour graph of Rosenbrock’s function shows a shallow banana shaped valley, it rep resents a significant challenge for gradient methods because of its shallowness, however the PSO method copes well with this function. Although convergence is more gradual for this function, the points are converging on the required single minimum.

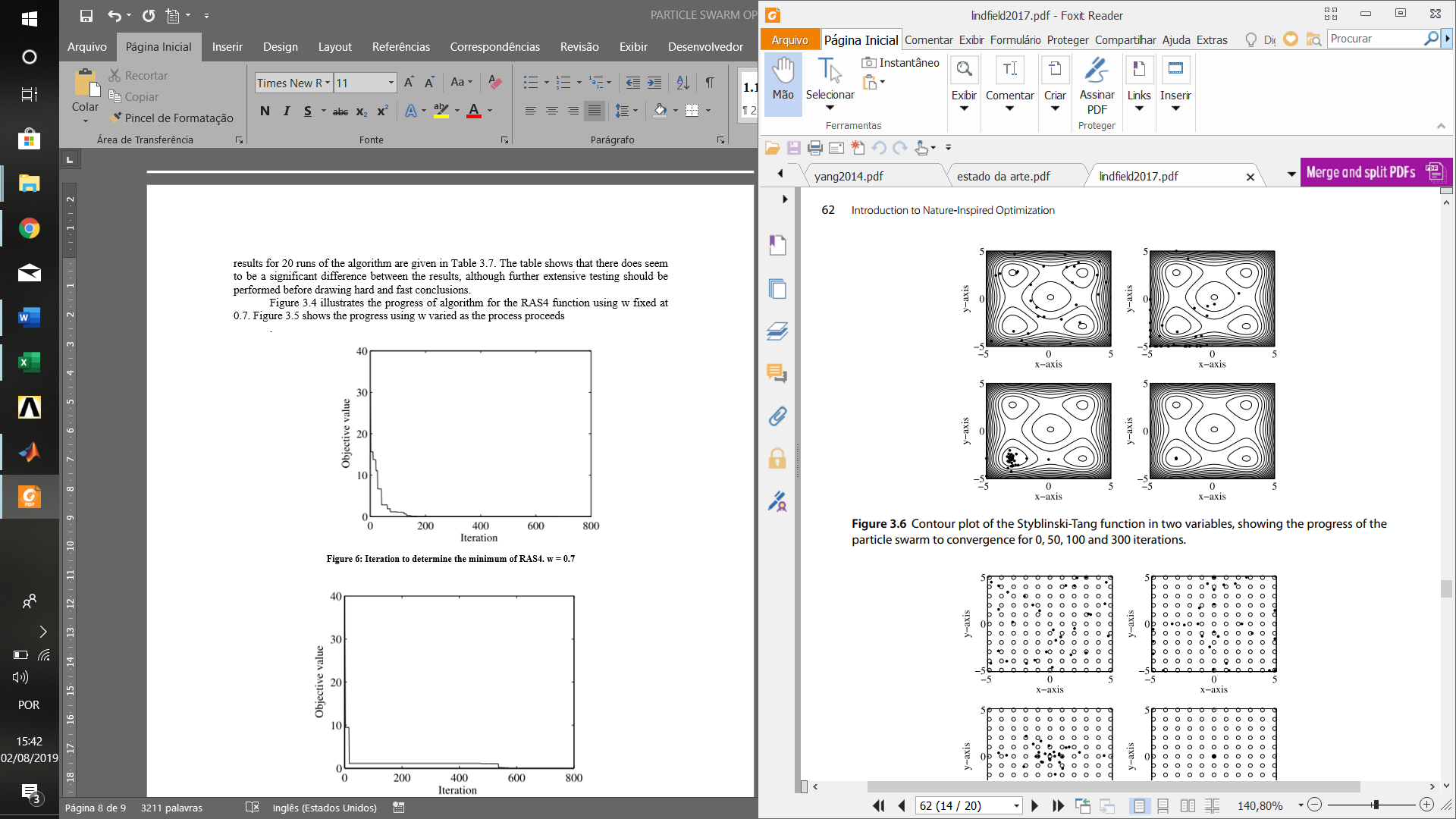


Figure 8: Contour plot of the Styblinski-Tang function in two variables, showing the progress of the particle swarm to convergence for 0, 50, 100 and 300 iterations

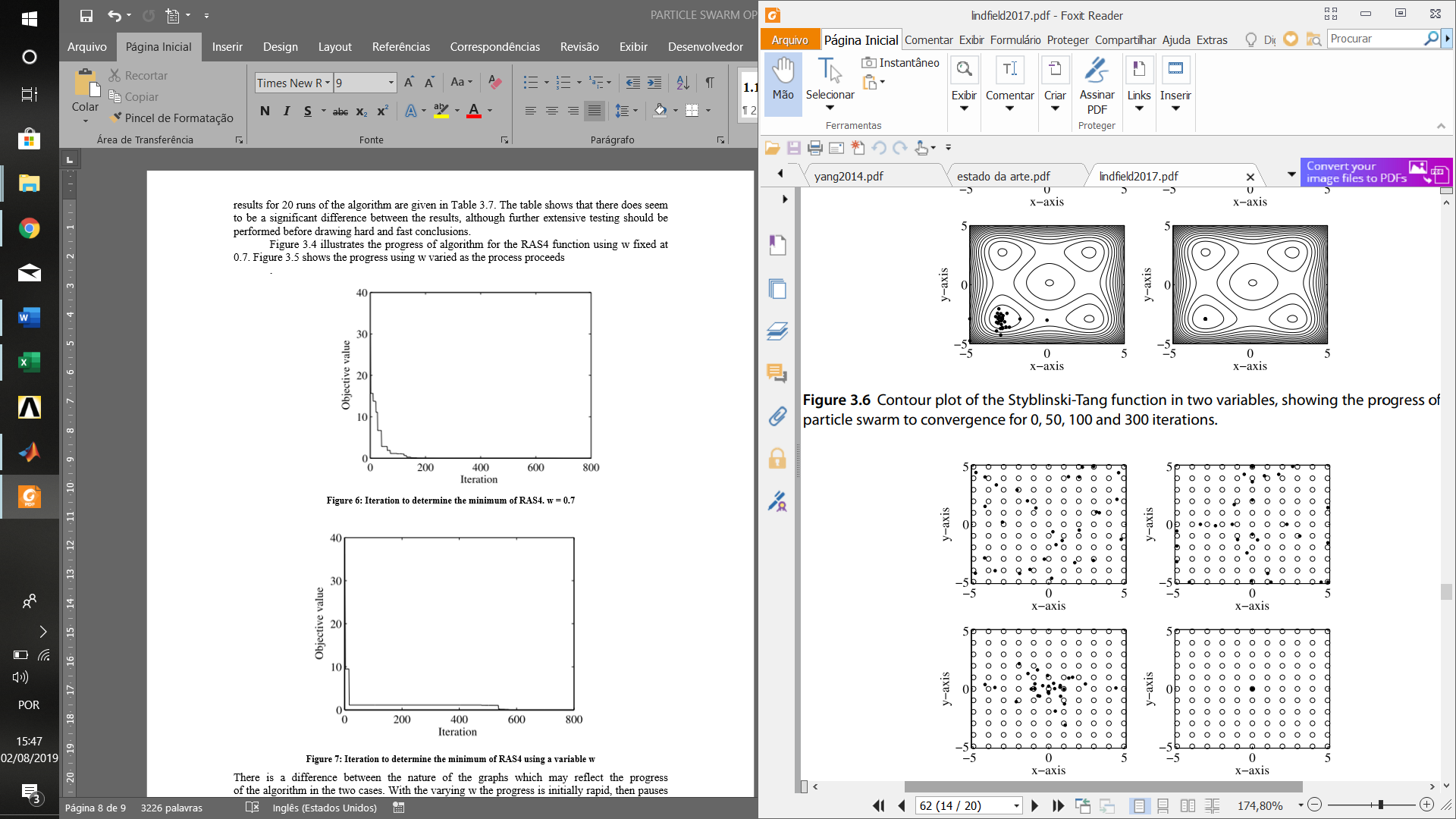


Figure 9: Rastrigin function in two variables, showing the progress of the particle swarm to convergence, for 0, 50, 100 and 300 iterations. The open circles show the many local minima.

MATLAB scripts have been used throughout to implement these tests and to generate the graphical representation.

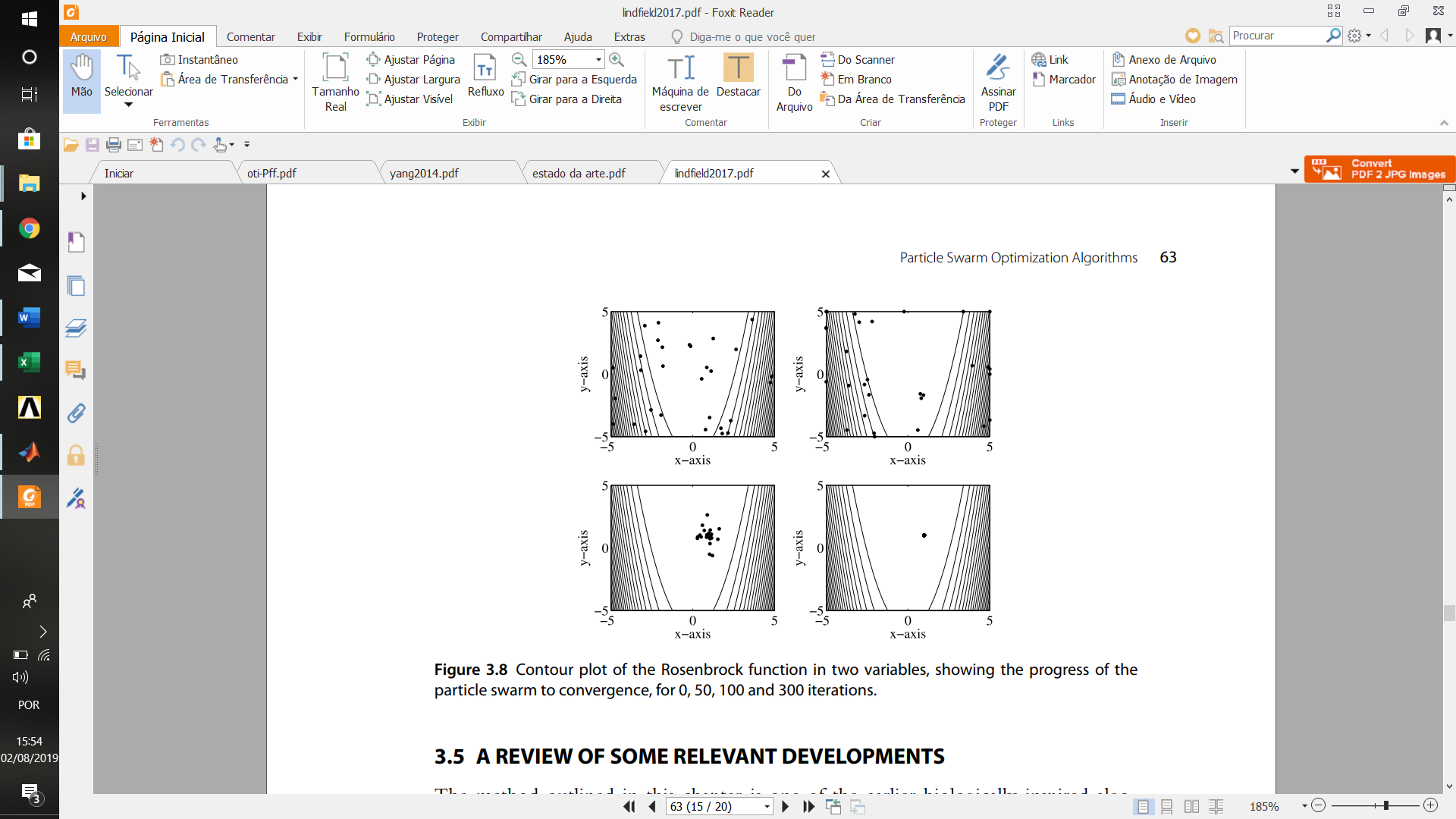


Figure 10: Contour plot of the Rosenbrock function in two variables, showing the progress of the particle swarm to convergence, for 0, 50, 100 and 300 iterations

* 1. PSO in civil structural engineering systems

References

[1] J. Kennedy e R. Eberhart, “Particle Swarm Optimization”, *G. Mal. Infett. Parassit.*, vol. 46, no 10, p. 1942–1948, 1994.

[2] K. Gopalakrishnan, *Particle Swarm Optimization in Civil Infrastructure Systems: State-of-the-Art Review*, First Edit. Elsevier Inc., 2013.

[3] J. T. Bryson, X. Jin, e S. K. Agrawal, “Optimal Design of Cable-Driven Manipulators Using Particle Swarm Optimization”, *J. Mech. Robot.*, vol. 8, no 4, p. 041003, 2015.

[4] X.-S. Yang, “Particle Swarm Optimization”, *Nature-Inspired Optim. Algorithms*, p. 99–110, 2014.