Escuela Superior de Cómputo
Instituto Politécnico Nacional
Ingeniería en Sistemas Computacionales

Evolutionary Computing

Practice 6: Particle Swarm Optimization

Professor: Jorge Luis Rosas Trigueros
Student: Ayala Segoviano Donaldo Horacio
Creation date: October 17th 2021
Delivery date: October 22th 2021

1 Introduction

1.1 Swarm Intelligence

Swarm intelligence consists of two or more individuals independently, or at least partially independently, acquire information and these different packages of information are combined and processed through social interaction, which provides a solution to a cognitive problem in a way that cannot be implemented by isolated individuals.

The main elements of of swarm intelligence are the following:

- **Group-living agents:** each agent can independently gather information and also copy the behaviour of others.
- **Learning capability:** agents are able to store and recall information. This ability could lead to cooperative interaction networks and potentially enable collective solutions.
- **Signalling:** it is possible for agents to inform others without physically guiding them.
- **Division of labour:** every individual is capable of adopting any task or role.

1.2 Particle Swarm Optimization

Particle Swarm Optimization uses the concept of swarm intelligence to solve optimization problems. It consists of particles that are flown through the problem hyperspace. When the population is initialized, in addition to the variables being given random values, they are stochastically assigned velocities. Each iteration, each particle's velocity is stochastically accelerated toward its previous best position (where it had its highest fitness value) and toward a neighborhood best position (the position of highest fitness by any particle in its neighborhood).

```
Loop
  For $i = 1$ to number of individuals
    if $G(\vec{x}_i) > G(\vec{p}_i)$ then do          //$G()$ evaluates fitness
      For $d = 1$ to dimensions
        $p_{id} = x_{id}$                             //$p_{id}$ is best so far
      Next $d$
    End do

    $g = i$                                           //arbitrary
    For $j$ = indexes of neighbors
      If $G(\vec{p}_j) > G(\vec{p}_g)$ then $g = j$     //$g$ is index of best performer
                                                      in the neighborhood

    Next $j$
    For $d = 1$ to number of dimensions
      $v_{id}(t) = v_{id}(t-1) + \varphi_1(p_{id} - x_{id}(t-1)) + \varphi_2(p_{gd} - x_{id}(t-1))$
      $v_{id} \in (-V_{\max}, +V_{\max})$
      $x_{id}(t) = x_{id}(t-1) + v_{id}(t)$
    Next $d$
  Next $i$
Until criterion
```

Figure 1.1: Pseudocode for PSO in continuous numbers.

Figure 1.1 shows the pseudocode for particle swarm optimization in continuous numbers. This algorithm will be used in this practice to optimize Ackley's and Rastrigin's functions in 2 and 3 dimensions respectively.

1.3 Optimization Functions

In applied mathematics, test functions, known as artificial landscapes, are useful to evaluate characteristics of optimization algorithms, such as:

- Convergence rate.
- Precision.

- Robustness.
- General performance.

In this practice optimization test functions will be useful to observe the efficiency of a Particle Swarm Optimization algorithms to find critical points in a search area. The ones used to test our genetic algorithms are *Ackley's* function using 2 dimensions and *Rastrigin's* function using 3 dimensions.

2 Material and equipment

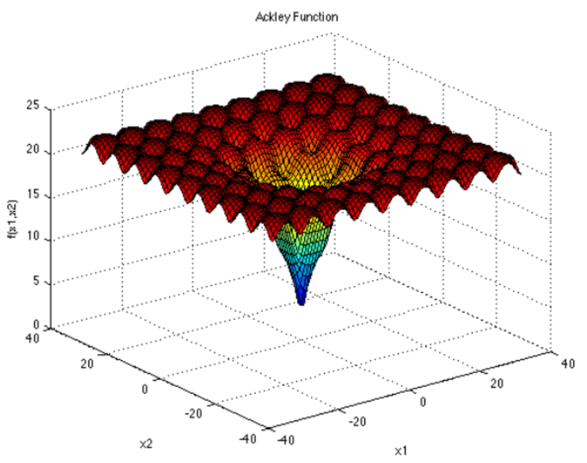
Following are the hardware and software used during the realization of this practice. ***Google Colaboratory*** was the tool used for the development of this practice and the next list shows the specifications of the hardware and software provided by Google Colab.

- **Hardware:**
  - **CPU:** Intel(R) Xeon(R) CPU @ 2.30GHz
  - **Memory:** 12GB
  - **Disk:** 108GB
- **Software:**
  - **Platform used:** *Google colaboratory* was used for this practice
  - **Programming language:** Python 3.7.11

3 Development

3.1 Optimization of Ackley's function

The algorithm shown in the introduction in figure 1.1 will be implemented in Python programming language to optimize Ackle's function which will be optimized in 2 dimensions. The objective will be to minimize the Ackley's function in two dimensions, so, the criteria to select the fittest particle will be to chose the one with the smallest result after evaluating a particle on the function.



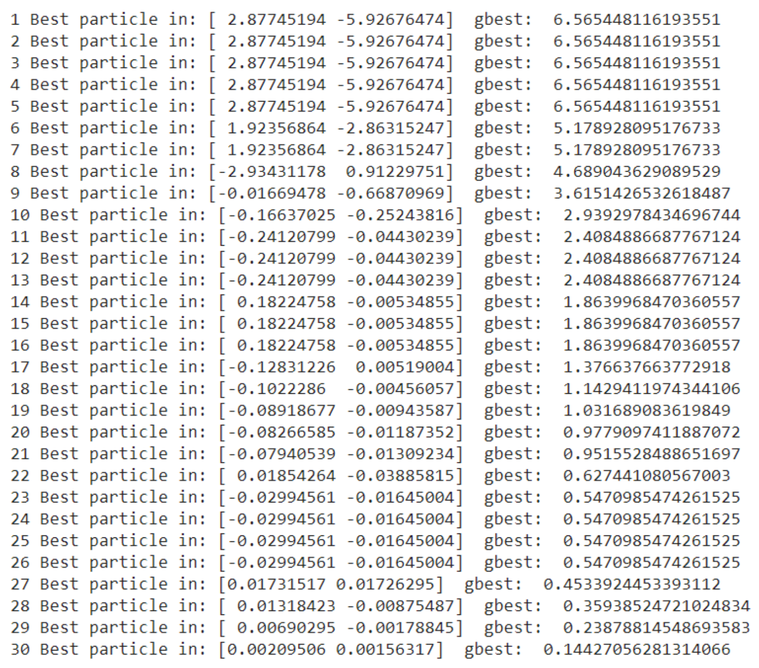
$$f(\mathbf{x}) = -a\exp\left(-b\sqrt{\frac{1}{d}\sum_{i=1}^{d}x_i^2}\right) - \exp\left(\frac{1}{d}\sum_{i=1}^{d}\cos(cx_i)\right) + a + \exp(1)$$

Figure 3.1: Generalization of Ackley's function and 2 dimensions plot.

Figure 3.1 shows the Ackley's function generalization formula and the plot for the function in 2 dimensions. It can be observed that the global minimum is the point $f(0,0) = 0$ and that it has several local minimums.

3.1.1 Implementation and testing

Once implemented, the program will be tested to verify that it produces a correct solution. Code implementation can be found by clicking **here**.

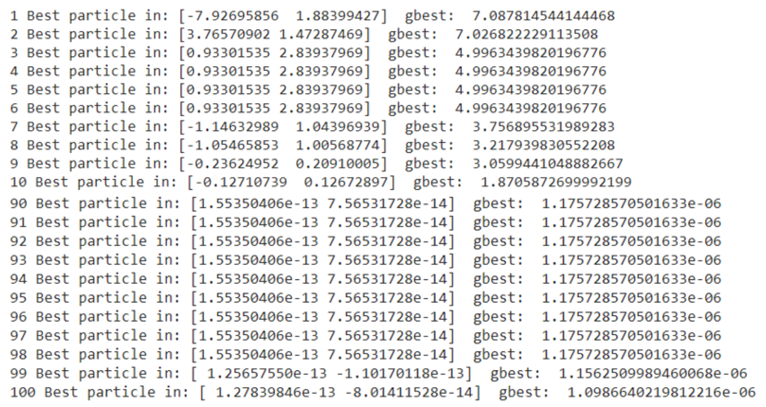


```
1 Best particle in: [ 2.87745194 -5.92676474]  gbest:  6.565448116193551
2 Best particle in: [ 2.87745194 -5.92676474]  gbest:  6.565448116193551
3 Best particle in: [ 2.87745194 -5.92676474]  gbest:  6.565448116193551
4 Best particle in: [ 2.87745194 -5.92676474]  gbest:  6.565448116193551
5 Best particle in: [ 2.87745194 -5.92676474]  gbest:  6.565448116193551
6 Best particle in: [ 1.92356864 -2.86315247]  gbest:  5.178928095176733
7 Best particle in: [ 1.92356864 -2.86315247]  gbest:  5.178928095176733
8 Best particle in: [-2.93431178  0.91229751]  gbest:  4.689043629089529
9 Best particle in: [-0.01669478 -0.66870969]  gbest:  3.6151426532618487
10 Best particle in: [-0.16637025 -0.25243816]  gbest:  2.9392978434696744
11 Best particle in: [-0.24120799 -0.04430239]  gbest:  2.4084886687767124
12 Best particle in: [-0.24120799 -0.04430239]  gbest:  2.4084886687767124
13 Best particle in: [-0.24120799 -0.04430239]  gbest:  2.4084886687767124
14 Best particle in: [ 0.18224758 -0.00534855]  gbest:  1.8639968470360557
15 Best particle in: [ 0.18224758 -0.00534855]  gbest:  1.8639968470360557
16 Best particle in: [ 0.18224758 -0.00534855]  gbest:  1.8639968470360557
17 Best particle in: [-0.12831226  0.00519004]  gbest:  1.376637663772918
18 Best particle in: [-0.1022286  -0.00456057]  gbest:  1.1429411974344106
19 Best particle in: [-0.08918677 -0.00943587]  gbest:  1.031689083619849
20 Best particle in: [-0.08266585 -0.01187352]  gbest:  0.9779097411887072
21 Best particle in: [-0.07940539 -0.01309234]  gbest:  0.9515528488651697
22 Best particle in: [ 0.01854264 -0.03885815]  gbest:  0.627441080567003
23 Best particle in: [-0.02994561 -0.01645004]  gbest:  0.5470985474261525
24 Best particle in: [-0.02994561 -0.01645004]  gbest:  0.5470985474261525
25 Best particle in: [-0.02994561 -0.01645004]  gbest:  0.5470985474261525
26 Best particle in: [-0.02994561 -0.01645004]  gbest:  0.5470985474261525
27 Best particle in: [0.01731517 0.01726295]  gbest:  0.4533924453393112
28 Best particle in: [ 0.01318423 -0.00875487]  gbest:  0.35938524721024834
29 Best particle in: [ 0.00690295 -0.00178845]  gbest:  0.23878814548693583
30 Best particle in: [0.00209506 0.00156317]  gbest:  0.14427056281314066
```

Figure 3.2: PSO's output for Ackley's function.

Figure 3.2 shows the output produced by the execution of the particle swarm optimization program for the Ackley's function. For this run, the search space was the cube of [-32, 32] in both dimensions, 30 iterations were done and 20 particles were used for each iteration. It can be observed that the final point is very close to the global minimum, giving a particle which is very close to $(0, 0)$ and a fitness value very close to 0.

Also, one can observe the evolution of the best solution found so far at each iteration, where the evalutaion of the best particle found went from over 6, to 0.14 and a a particle very close to the global minimum coordinates. Now, the resources will be incremented to see how much improvement there is.



```
1 Best particle in: [-7.92695856  1.88399427]  gbest:  7.087814544144468
2 Best particle in: [3.76570902 1.47287469]  gbest:  7.026822229113508
3 Best particle in: [0.93301535 2.83937969]  gbest:  4.9963439820196776
4 Best particle in: [0.93301535 2.83937969]  gbest:  4.9963439820196776
5 Best particle in: [0.93301535 2.83937969]  gbest:  4.9963439820196776
6 Best particle in: [0.93301535 2.83937969]  gbest:  4.9963439820196776
7 Best particle in: [-1.14632989  1.04396939]  gbest:  3.756895531989283
8 Best particle in: [-1.05465853  1.00568774]  gbest:  3.217939830552208
9 Best particle in: [-0.23624952  0.20910005]  gbest:  3.0599441048882667
10 Best particle in: [-0.12710739  0.12672897]  gbest:  1.8705872699992199
90 Best particle in: [1.55350406e-13 7.56531728e-14]  gbest:  1.175728570501633e-06
91 Best particle in: [1.55350406e-13 7.56531728e-14]  gbest:  1.175728570501633e-06
92 Best particle in: [1.55350406e-13 7.56531728e-14]  gbest:  1.175728570501633e-06
93 Best particle in: [1.55350406e-13 7.56531728e-14]  gbest:  1.175728570501633e-06
94 Best particle in: [1.55350406e-13 7.56531728e-14]  gbest:  1.175728570501633e-06
95 Best particle in: [1.55350406e-13 7.56531728e-14]  gbest:  1.175728570501633e-06
96 Best particle in: [1.55350406e-13 7.56531728e-14]  gbest:  1.175728570501633e-06
97 Best particle in: [1.55350406e-13 7.56531728e-14]  gbest:  1.175728570501633e-06
98 Best particle in: [1.55350406e-13 7.56531728e-14]  gbest:  1.175728570501633e-06
99 Best particle in: [ 1.25657550e-13 -1.10170118e-13]  gbest:  1.1562509989460068e-06
100 Best particle in: [ 1.27839846e-13 -8.01411528e-14]  gbest:  1.0986640219812216e-06
```

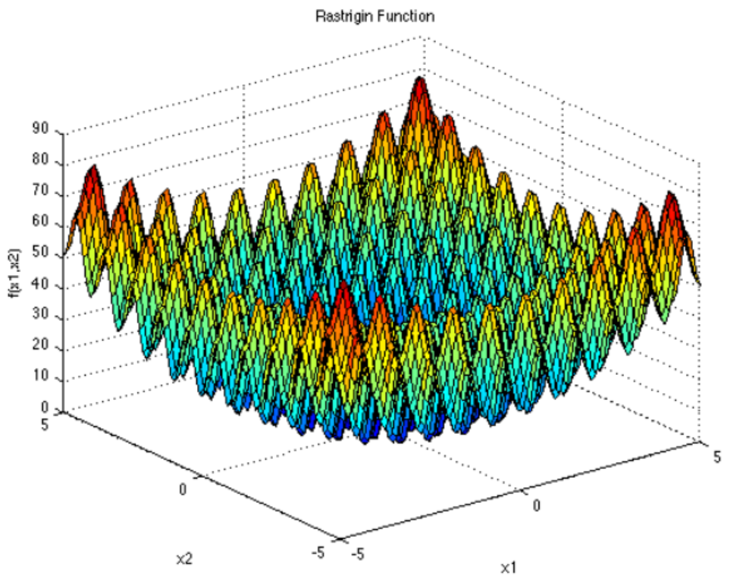
Figure 3.3: PSO's with increased resources output for Ackley's function.

Figure 3.3 shows the output of the execution of the program with increased resources, for this run, the same search space was used, however, this time, 100 particles were used and 100 iterations were done. It can be seen in the screenshot that the accuracy increased considerably, the points are way closer to the global minimum and the objective function is way smaller.

This way, it can be concluded that the algorithm used is quite efficient for optimizing this function, since for low resources (number of particles and iterations) the result found was very close to the global minimum.

3.2 Optimization of Rastrigin's function

In this section, the algorithm mentioned in the introduction for particle swarm optimization will be implemented and used to minimize the Rastrigin's function in 3 dimensions. The criteria to select the fittest particles is going to be selecting the particles with the smallest output after evaluating them on the Rastrigin's function.



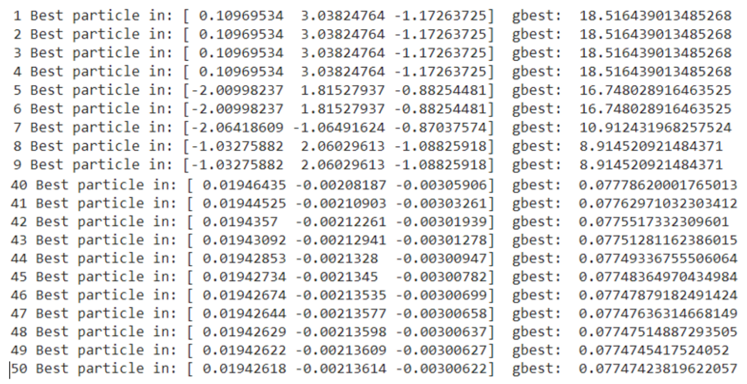
$$f(\mathbf{x}) = 10d + \sum_{i=1}^{d} \left[x_i^2 - 10\cos(2\pi x_i)\right]$$

Figure 3.4: Generalization of Rastrigin's function and 2 dimensions plot.

Figure 3.4 shows the generalization of the Rastrigin's function, where d is the number of dimensions, in this case, it will be 3. It can be observed that the global minimum is located in $f(0,0,0) = 0$ and that it has many local minimum points.

3.2.1 Implementation and testing

After implementing the program in Python programming language, it is now going to be tested to verify the efficacy and efficiency of the program. The implementation can be found by clicking **here**.

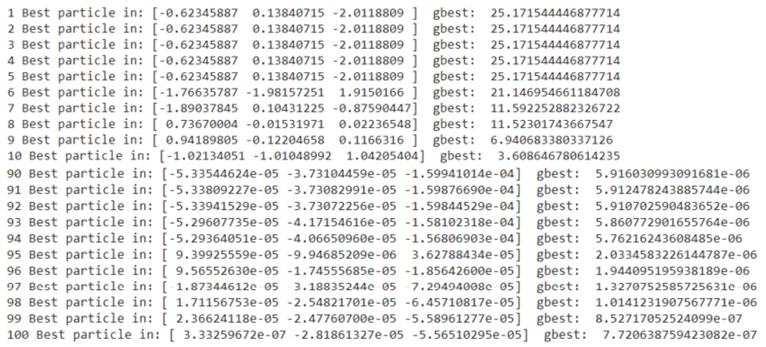


```
 1 Best particle in: [ 0.10969534  3.03824764 -1.17263725]  gbest:  18.516439013485268
 2 Best particle in: [ 0.10969534  3.03824764 -1.17263725]  gbest:  18.516439013485268
 3 Best particle in: [ 0.10969534  3.03824764 -1.17263725]  gbest:  18.516439013485268
 4 Best particle in: [ 0.10969534  3.03824764 -1.17263725]  gbest:  18.516439013485268
 5 Best particle in: [-2.00998237  1.81527937 -0.88254481]  gbest:  16.748028916463525
 6 Best particle in: [-2.00998237  1.81527937 -0.88254481]  gbest:  16.748028916463525
 7 Best particle in: [-2.06418609 -1.06491624 -0.87037574]  gbest:  10.912431968257524
 8 Best particle in: [-1.03275882  2.06029613 -1.08825918]  gbest:  8.914520921484371
 9 Best particle in: [-1.03275882  2.06029613 -1.08825918]  gbest:  8.914520921484371
40 Best particle in: [ 0.01946435 -0.00208187 -0.00305906]  gbest:  0.07778620001765013
41 Best particle in: [ 0.01944525 -0.00210903 -0.00303261]  gbest:  0.07762971032303412
42 Best particle in: [ 0.0194357  -0.00212261 -0.00301939]  gbest:  0.0775517332309601
43 Best particle in: [ 0.01943092 -0.00212941 -0.00301278]  gbest:  0.07751281162386015
44 Best particle in: [ 0.01942853 -0.0021328  -0.00300947]  gbest:  0.07749336755506064
45 Best particle in: [ 0.01942734 -0.0021345  -0.00300782]  gbest:  0.07748364970434984
46 Best particle in: [ 0.01942674 -0.00213535 -0.00300699]  gbest:  0.07747879182491424
47 Best particle in: [ 0.01942644 -0.00213577 -0.00300658]  gbest:  0.07747636314668149
48 Best particle in: [ 0.01942629 -0.00213598 -0.00300637]  gbest:  0.07747514887293505
49 Best particle in: [ 0.01942622 -0.00213609 -0.00300627]  gbest:  0.0774745417524052
50 Best particle in: [ 0.01942618 -0.00213614 -0.00300622]  gbest:  0.07747423819622057
```

Figure 3.5: PSO's output for Rastrigin's function.

Figure 3.5 shows the execution of the particle optimization program for the Rastrigin's function in three dimensions. For this run, the space search used was a cube with boundaries of $[-6, 6]$ in each dimension, also, since there are more dimensions for this function and the Rastrigin's function is harder to optimize, in the sense that it has many local minimums and their values are more similar than with the Ackley's function, more particles and iterations were used for this first run, using 50 particles and 50 iterations seems to have a good outcome, it can be seen in the screenshot that the evolution is quite good, since it starts with a fitness of 18 and it decreases to 0.07, which is really close to the global minimum.

In order to see how more accurate it can be, the parameters will be increased, i.e. the number of particles and the number of iterations, so we can verify the improvements compared to the last run.



```
1 Best particle in: [-0.62345887  0.13840715 -2.0118809 ]  gbest:  25.171544446877714
2 Best particle in: [-0.62345887  0.13840715 -2.0118809 ]  gbest:  25.171544446877714
3 Best particle in: [-0.62345887  0.13840715 -2.0118809 ]  gbest:  25.171544446877714
4 Best particle in: [-0.62345887  0.13840715 -2.0118809 ]  gbest:  25.171544446877714
5 Best particle in: [-0.62345887  0.13840715 -2.0118809 ]  gbest:  25.171544446877714
6 Best particle in: [-1.76635787 -1.98157251  1.9150166 ]  gbest:  21.146954661184708
7 Best particle in: [-1.89037845  0.10431225 -0.87590447]  gbest:  11.592252882326722
8 Best particle in: [ 0.73670004 -0.01531971  0.02236548]  gbest:  11.52301743667547
9 Best particle in: [ 0.94189805 -0.12204658  0.1166316 ]  gbest:  6.940683380337126
10 Best particle in: [-1.02134051 -1.01048992  1.04205404]  gbest:  3.608646780614235
90 Best particle in: [-5.33544624e-05 -3.73104459e-05 -1.59941014e-04]  gbest:  5.916030993091681e-06
91 Best particle in: [-5.33809227e-05 -3.73082991e-05 -1.59876690e-04]  gbest:  5.912478243885744e-06
92 Best particle in: [-5.33941529e-05 -3.73072256e-05 -1.59844529e-04]  gbest:  5.910702590483652e-06
93 Best particle in: [-5.29607735e-05 -4.17154616e-05 -1.58102318e-04]  gbest:  5.860772901655764e-06
94 Best particle in: [-5.29364051e-05 -4.06650960e-05 -1.56806903e-04]  gbest:  5.76216243608485e-06
95 Best particle in: [ 9.39925559e-05 -9.94685209e-06  3.62788434e-05]  gbest:  2.0334583226144787e-06
96 Best particle in: [ 9.56552630e-05 -1.74555685e-05 -1.85642600e-05]  gbest:  1.944095195938189e-06
97 Best particle in: [-1.87344612e-05 -3.18835244e-05 -7.29494008e-05]  gbest:  1.3270752585725631e-06
98 Best particle in: [ 1.71156753e-05 -2.54821701e-05 -6.45710817e-05]  gbest:  1.0141231907567771e-06
99 Best particle in: [ 2.36624118e-05 -2.47760700e-05 -5.58961277e-05]  gbest:  8.52717052524099e-07
100 Best particle in: [ 3.33259672e-07 -2.81861327e-05 -5.56510295e-05]  gbest:  7.720638759423082e-07
```

Figure 3.6: PSO's output for Rastrigin's function with increased parameters.

In figure 3.6 we can observe the first and the last 10 iterations of the execution, and it is clear that a considerable improvement was made compared to the last run, and also the difference between the initial iteration and the last best particle found is quite large, so the evolution is also quite good.

After analysing the results, it can be concluded that the algorithm for finding the global minimum of the Rastrigin's function is efficient and it can be able to find fairly good solutions. And even though compared to the Ackley's function, more resources in terms of particles and iterations were used to have acceptable results, the amount of resources used for this program were also pretty low.

4 Conclusions

The development of his practice resulted in a better and deeper understanding of Particle Swarm Optimization, and it could be understood how the concept of Swarm intelligence can be applied to solve problems, in this case of optimization.

It was concluded that Particle Swarm Optimization is a quite good and efficient algorithm for optimizing functions, it could be observed in the results obtained, that the results were very accurate and close to the global.

Also, it could be observed that the Particle Swarm Optimization algorithm is easier to implements and to adapt to different problems, compared to genetic algorithms (for continuous function optimization), where for each problem, it was necessary to think of a way to represent chromosomes and evaluate them as well as defining mutation.

5 Bibliography

- **Kennedy J., C. Eberhart R. (2001) Swarm Intelligence. San Diego, Ca, USA. Academic Press, Morgan Kaufmann.**
- **Krause J., Ruxton G., Krause S. (2010). Swarm intelligence in animals and humans. Trends in Ecology Evolution.**
- **Surjanovic Sonja, Bingham Derek (2013). Virtual Library of Simulation Experiments. Burnaby, Cánada. Simon Fraser University.**