22

Social Spider Optimization

Ahmed F. Ali
Department of Computer Science
Suez Canal University, Ismaillia, Egypt

Mohamed A. Tawhid
Department of Mathematics and Statistics
Faculty of Science, Thompson Rivers University, Kamloops, Canada

CONTENTS

22.1 Introduction

Social spider optimization algorithm (SSO) is a recent population based swarm intelligence algorithm proposed by Cuevas et al. [4]. The SSO algorithm was inspired by the social behavior of the social spider colony that consists of social members and a communal web. The SSO has been applied in many applications due to its efficiency especially when it is applied to solve global optimization problems [5]. The authors in [8] have combined the SSO algorithm and Nelder-Mead search method to solve minimax and integer programming problems. The authors in [3] have applied the SSO algorithm for text document clustering, while the authors in [9] have proposed a new hybrid SSO and genetic algorithm to minimize the potential energy function and large scale optimization problems. The rest of the chapter is organized as follows. In Section 22.2, we present the standard SSO algorithm. In Sections 22.3 and 22.4, we give Matlab and C++ source codes of the SSO algorithm, respectively. In Section 22.5, we illustrate a step by step numerical example of the SSO algorithm. Finally, we outline the conclusion of this chapter in Section 22.6.

22.2 Original SSO algorithm

In the following subsections, we describe the main concepts of the SSO algorithm and how it works.

22.2.1 Social behavior and inspiration

The SSO algorithm mimics the social behavior of the social spider colony that consists of social members and a communal web. The social members are divided into males and females. The number of female and male spiders reaches to 70% and 30% of the total colony members [1], [2], respectively. Each member in the colony is responsible for a specific task such as building and maintaining the communal web, prey capturing, and mating [6]. The female spiders produce an attraction or dislike over others. The vibrations in the communal web are based on the weight and distance of the members which are the main features of the attraction or dislike of a particular spider [7]. Male spiders have two categories, dominant and non-dominant [10]. The dominant male spiders have better fitness characteristics than the non-dominant spiders. The dominant male can mate with one or all females in the colony to exchange information among members and produce offspring. In the social spider optimization algorithm (SSO), each solution represents a spider position, while the communal web represents the search space. The value of each solution is represented by calculating its fitness function which represents the weight of each spider.

22.2.2 Population initialization

In the SSO algorithm, the population consists of N solutions (spiders) and can be divided into females f_i and males m_i. The number of females N_f is selected within the range of $65\% - 90\%$ and it can be calculated by the following equation:

$$N_f = floor[(0.9 - rand(0,1) \cdot 0.25) \cdot N] \tag{22.1}$$

where rand is a random number between (0,1) and floor(.) converts a real number to an integer number. The number of male spiders N_m can be calculated as follows.

$$N_m = N - N_f \tag{22.2}$$

The female spider position f_i is randomly generated within the lower p_j^{low} and the upper p_j^{high} initial parameter bounds as follows.

$$\begin{gathered} f_{i,j}^0 = p_j^{low} + rand(0,1).(p_j^{high} - p_j^{low}) \\ i = 1, 2, \ldots, N_f; j = 1, 2, \ldots, n \end{gathered} \tag{22.3}$$

The male spider position m_i is randomly generated as follows.

$$\begin{gathered} m_{k,j}^0 = p_j^{low} + rand(0,1).(p_j^{high} - p_j^{low}) \\ k = 1, 2, \ldots, N_m; j = 1, 2, \ldots, n \end{gathered} \tag{22.4}$$

The zero signals represent the initial population and j, i and k are the parameter and individual indices, respectively. The value of $f_{i,j}$ is the jth parameter of the ith female spider position.

22.2.3 Evaluation of the solution quality

In the SSO algorithm, the solution quality is represented by the weight of each spider. Each solution i is evaluated by calculating its fitness function value as follows.

$$w_i = \frac{J(s_i) - worst_s}{best_s - worst_s} \tag{22.5}$$

where $J(s_i)$ is the fitness value of the spider position s_i with regard to the substituted objective function $J(\cdot)$. The value $worst_s$ represents the maximum solution's value while the $best_s$ represents the minimum value of the solution in the population. These values are defined by considering the following minimization problem as follows.

$$best_s = \min_{k \in \{1,\ldots,N\}} J(s_k) \quad \text{and } worst_s = \max_{k \in \{1,\ldots,N\}} J(s_k) \tag{22.6}$$

22.2.4 Modeling of the vibrations through the communal web

The colony members share and transmit their information through the communal web by encoding it as small vibrations. These vibrations are critical for the collective coordination of all members in the population. The weight and the distance of the spider are responsible for generating these vibrations. The transmitted information (vibrations) perceived by the solution i from solution j is modeled as follows.

$$Vib_{i,j} = w_j \cdot e^{-d_{i,j}^2} \tag{22.7}$$

where the $d_{i,j}$ is the Euclidian distance between the spiders i and j.

The vibrations between any pair of individuals can be defined as follows.

- **Vibrations** $Vibc_i$. The vibrations (transmitted information) between the solution i and the nearest solution to it (which is solution c (s_c) that has a higher weight) can be defined as follows.

$$Vibc_i = w_c \cdot e^{-d_{i,c}^2} \tag{22.8}$$

- **Vibrations** $Vibb_i$. The vibrations (transmitted information) between the solution i and the best solution b (s_b) in the population can be defined as follows.

$$Vibb_i = w_b \cdot e^{-d_{i,b}^2} \tag{22.9}$$

- **Vibrations** $Vibf_i$. Finally, the vibrations (transmitted information) between the solution i and the nearest female solution $f(s_f)$ can be defined as

$$Vibf_i = w_f \cdot e^{-d_{i,f}^2} \tag{22.10}$$

22.2.5 Female cooperative operator

The female spiders attract or dislike other males. The movement of attraction or repulsion based on several random phenomena. A uniform random number r_m is generated within the range [0,1]. The attraction movement is generated if r_m is smaller than a threshold PF, otherwise, a repulsion movement is produced as follows.

$$f_i^{t+1} = \begin{cases} f_i^t + \alpha \cdot Vibc_i \cdot (s_c - f_i^t) + \beta \cdot Vibb_i \cdot (s_b - f_i^t) \\ \qquad +\delta \cdot (rand - 0.5) \text{ at } PF \\ f_i^t - \alpha \cdot Vibc_i \cdot (s_c - f_i^t) - \beta \cdot Vibb_i \cdot (s_b - f_i^t) \\ \qquad +\delta \cdot (rand - 0.5) \text{ at } 1 - PF \end{cases} \tag{22.11}$$

where α, β, δ and rand are random numbers in [0,1], whereas t is the number of iterations.

22.2.6 Male cooperative operator

The dominant male spider D is the spider with a weight value above the median value of the other males in the population, while the other males with weights under the median are called non-dominant ND. The median weight is indexed by $N_f + m$. The position of the male spider can be defined as the following.

$$m_i^{t+1} = \begin{cases} m_i^t + \alpha \cdot Vibf_i \cdot (s_f - m_i^t) + \delta \cdot (rand - 0.5) & \text{if} \quad w_{N_f+i} > w_{N_f+m} \\ m_i^t + \alpha \cdot \left(\frac{\sum_{h=1}^{N_m} m_h^t \cdot w_{N_f+h}}{\sum_{h=1}^{N_m} w_{N_f+h}} - m_i^t \right) \end{cases} \tag{22.12}$$

where the solution s_f represents the nearest female solution to the male solution i.

22.2.7 Mating operator

The dominant male is responsible for mating a set E^g of female members when it locates them within a specific range r (range of mating), which can be calculated as follows.

$$r = \frac{\sum_{j=1}^{n} (p_j^{high} - p_j^{low})}{2.n} \tag{22.13}$$

The spider with a heavier weight has a big chance to influence the new product. The influence probability Ps_i of each solution is assigned by the roulette wheel selection method as follows.

$$Ps_i = \frac{w_i}{\sum_{j \in T^t} w_j} \tag{22.14}$$

22.2.8 Pseudo-code of SSO algorithm

In this subsection, we present the pesudo-code of the SSO algorithm as shown in Algorithm 20.

Algorithm 20 Social spider optimization algorithm.

Set the initial value of total number of solutions N in the population size S, threshold PF, and maximum number of iterations Max_{itr}
Set the number of female spiders N_f and number of males spiders N_m as in (22.1) and (22.2)
Set $t := 0$ ▷ **Counter initialization**
for $(i = 1; i < N_f + 1; i++)$ **do**
  for $(j = 1; j < n + 1; j++)$ **do**
    $f_{i,j}^t = p_j^{low} + rand(0,1) \cdot (P_j^{high} - p_j^{low})$
  end for
end for ▷ **Initialize randomly the female spider**
for $(k = 1; k < N_m + 1; k++)$ **do**
  for $(j = 1; j < n + 1; j++)$ **do**
    $m_{k,j}^t = p_j^{low} + rand(0,1) \cdot (P_j^{high} - p_j^{low})$
  end for
end for ▷ **Initialize randomly the male spider**
repeat

for $(i = 1; i < N + 1; i + +)$ **do**
$w_i = \frac{J(s_i) - worst_s}{best_s - worst_s}$
end for ▷ **Evaluate the weight (fitness function) of each spider**
for $(i = 1; i < N_f + 1; i + +)$ **do**
Calculate the vibrations of the best local and best global solutions $Vibc_i$ and $Vibb_i$ as in (22.8) and (22.9)
if $(r_m < PF)$ **then**
$f_i^{t+1} = f_i^t + \alpha \cdot Vibc_i \cdot (s_c - f_i^t) + \beta \cdot Vibb_i \cdot (s_b - f_i^t) + \delta \cdot (rand - 0.5)$
else
$f_i^{t+1} = f_i^t - \alpha \cdot Vibc_i \cdot (s_c - f_i^t) - \beta \cdot Vibb_i \cdot (s_b - f_i^t) + \delta \cdot (rand - 0.5)$
end if
end for
Find the median male individual (w_{N_f+m}) from M
for $(i = 1; i < N_m + 1; i + +)$ **do**
Calculate $Vibf_i$ as in (22.10)
if $(w_{N_f i} > w_{N_f+m})$ **then**
$m_i^{t+1} = m_i^t + \alpha \cdot Vibf_i \cdot (s_f - m_i^t) + \delta \cdot (rand - 0.5)$
else
$m_i^{t+1} = m_i^t + \alpha \cdot \left(\frac{\sum_{h=1}^{N_m} m_h^t \cdot w_{N_f+h}}{\sum_{h=1}^{N_m} w_{N_f+h}} - m_i^t \right)$
end if
end for
Calculate the radius of mating r, where $r = \frac{\sum_{j=1}^{n} (p_i^{high} - p_j^{low})}{2 \cdot n}$ ▷ **Perform the mating operation**
for $(i = 1; i < N_m + 1; i + +)$ **do**
if $(m_i \in D)$ **then**
Find E^i
if E^i is not empty **then**
Form s_{new} using the roulette method
if $w_{new} > w_{wo}$ **then**
Set $s_{wo} = s_{new}$
end if
end if
end if
end for
$t = t + 1$ ▷ **Iteration counter is increasing**
until $(t \geq Max_{itr})$ ▷ **Termination criteria are satisfied**
Produce the best solution

22.2.9 Description of the SSO algorithm

In this subsection, we give a description of the SSO algorithm. The algorithm initializes the values of the number of solutions N in the population size P, threshold PF and the maximum number of iterations Max_{itr}. The number of female and male solutions are assigned as shown in (22.1) and (22.2). The counter of the initial iteration is initialized and the initial population is randomly generated which contains the female and the male solutions. The following steps are repeated until termination criteria are satisfied.

- The fitness function of each solution is calculated to determine its weight as shown in (22.5).

- The female spiders are moving according to their cooperative operator by calculating the vibrations of the local and global best spiders as shown in (22.8) and (22.9).
- The male spiders are moving according to their cooperative operator by calculating the median male solution w_{N_f+m} from all male spiders.
- The mating operation is applied by calculating the radius of mating as shown in (22.13).
- The number of iterations is increased.

The overall process is repeated until termination criteria are satisfied. Finally, the best obtained solution is presented as the optimal or near optimal solution.

22.3 Source-code of SSO algorithm in Matlab

In this section, we present the source code of the fitness function, which we need to minimize by using SSO algorithm as shown in Listing 22.1. The fitness function is shown in Equation 22.15. The function takes the whole population X and outputs the objective function of each solution in the population. Also, we present the source codes of the main SSO algorithm and its functions in Matlab [4] as shown in Listing 22.2.

$$f(X_i) = \sum_{j=1}^{D} X_{i,j}^2 \qquad \text{where } -10 \leqslant X_{i,j} \leqslant 10 \tag{22.15}$$

```
function [out]=fun(X)
[x,y]=size(X);
out=zeros(x,1);
for i=1:x
    for j=1:y
    out(i,1)=out(i,1)+X(i,j)^2;
    end
end
```

Listing 22.1
Definition of objective function $fun(.)$ in Matlab.

```
function [bfit,befit,spbest,spbesth] = SSO(spidn,L,U,fun,dim,itern)

     f=fun;
     xd=L;
  xu=U;
     dims = dim;
  for i=1:dims
          lb(i,:)=xd;
          ub(i,:)=xu;
     end
     rand(state,0);  % Reset the random generator
```

```
    % Define the population of females and males
    fpl = 0.65;      % Lower Female Percent
    fpu = 0.9;       % Upper Female Percent
    fp = fpl+(fpu-fpl)\cdot rand; % Aleatory Percent
    fn = round(spidn\cdot fp);    % Number of females
    mn = spidn-fn;          % Number of males
  %Probabilities of attraction or repulsion
  % Proper tuning for better results
    pm = exp(-(0.1:(3-0.1)/(itern-1):3));
    % Initialization of vectors
    fsp = zeros(fn,dims);    % Initialize females
    msp = zeros(mn,dims);    % Initialize males
    fefit = zeros(fn,1);     % Initialize fitness females
    mafit = zeros(mn,1);     % Initialize fitness males
    spwei = zeros(spidn,1); % Initialize weigth spiders
    fewei = zeros(fn,1); % Initialize weigth spiders
    mawei = zeros(mn,1); % Initialize weigth spiders
    %% Population Initialization
    % Generate Females
    for i=1:fn
        fsp(i,1:dims)=lb(1)+rand(1,dims)* (ub(1)-lb(1));
        fsp=round(fsp);
    end
    % Generate Males
    for i=1:mn
        msp(i,1:dims)=lb(1)+rand(1,dims).* (ub(1)-lb(1));
        msp=round(msp);
    end
    %% **** Evaluations *****
  % Evaluation of function for females
    for i=1:fn
        fefit(i)=f(fsp(i,:),dims);
    end
  % Evaluation of function for males
    for i=1:mn
        mafit(i)=f(msp(i,:),dims);
    end
    %% ***** Assign weight or sort ***********
  % Obtain weight for every spider
    spfit = [fefit mafit];    % Mix Females and Males
    bfitw = min(spfit);           % best fitness
    wfit = max(spfit);            % worst fitness
    for i=1:spidn
        spwei(i) = 0.001+((spfit(i)-wfit)/(bfitw-wfit));
    end
    fewei = spwei(1:fn);        % Separate the female mass
    mawei = spwei(fn+1:spidn);% Separate the male mass
    %% Memory of the best
    % Check the best position
    [~,Ibe] = max(spwei);
    % Check if female or male
    if Ibe > fn
        % Is Male
        spbest=msp(Ibe-fn,:);    % Assign best position to spbest
        bfit = mafit(Ibe-fn);        % Get best fitness for memory
    else
        % Is Female
        spbest=fsp(Ibe,:);        % Assign best position to spbest
        bfit = fefit(Ibe);        % Get best fitness for memory
    end
    %% Start the iterations
    for i=1:itern
     %% ***** Movement of spiders *****
     % Move Females
     [fsp] = FeMove(spidn,fn,fsp,msp,spbest,Ibe,spwei,dims,lb,ub,pm(i)
    );
     % Move Males
```

```
        [msp] = MaMove(fn,mn,fsp,msp,fewei,mawei,dims,lb,ub,pm(i));
        %% **** Evaluations *****
        % Evaluation of function for females
        for j=1:fn
            fefit(j)=f(fsp(j,:),dims);
        end
        % Evaluation of function for males
        for j=1:mn
            mafit(j)=f(msp(j,:),dims);
        end

end
```

Listing 22.2
The main code for the social spider optimization algorithm $SSO(.)$ in Matlab.

The rest of the Matlab code is presented in https://www.mathworks.com/matlabcentral/fileexchange/ 46942-a-swarm--optimization-algorithm-inspired-in-the-behavior-of-the-social-spider

22.4 Source-code of SSO algorithm in C++

In this section, we highlight the C++ code of SSO algorithm as follows.

```
#include "SSA.h"

using namespace std;

class fun : public Problem {
public:
  fun(unsigned int dimension) : Problem(dimension) { }

  double eval(const std::vector<double>& solution) {
  double sum = 0.0;
  for (int i = 0; i < solution.size(); ++i) {
      sum += solution[i] * solution[i];
      }
    return sum;
    }
```

Listing 22.3
Definition of objective function $fun(.)$ and the main file in C++.

```
#ifndef SSA_SSA_H
#define SSA_SSA_H

#include <chrono>
#include <iostream>
#include <math.h>
#include <stdio.h>
#include <stdlib.h>
#include <vector>

class Problem {
public:
    unsigned int dimension;

    Problem(unsigned int dimension) : dimension(dimension) { }
```

```cpp
    virtual double eval(const std::vector<double>& solution) = 0;
};

class Position {
public:
    double fitness;
    std::vector<double> solution;

    Position() { };
    Position(const std::vector<double>& solution, double fitness) :
            solution(solution), fitness(fitness) { }

    friend bool operator==(const Position& p1, const Position& p2) {
        for (int i = 0; i < p1.solution.size(); ++i) {
            if (p1.solution[i] != p2.solution[i]) {
                return false;
            }
        }
        return true;
    }

    friend double operator-(const Position& p1, const Position& p2) {
        double distance = 0.0;
        for (int i = 0; i < p1.solution.size(); ++i) {
            distance += fabs(p1.solution[i] - p2.solution[i]);
        }
        return distance;
    }

    static Position init_position(Problem* problem);
};

class Vibration {
public:
    double intensity;
    Position position;
    static double C;

    Vibration() { }
    Vibration(const Position& position);
    Vibration(double intensity, const Position& position);

    double intensity_attenuation(double attenuation_factor, double
    distance) const;
    static double fitness_to_intensity(double fitness);
};
```

Listing 22.4
SSA header file in C++.

The rest of the social spider optimization C++ code is represented in https://github.com/James-Yu/SocialSpiderAlgorithm.

22.5 Step-by-step numerical example of SSO algorithm

In this section, we apply the steps of the SSO algorithm in Algorithm 20 to minimize the function in Equation 22.15. We set the initial population size $N = 6$ and the maximum number of iterations Max_{itr} in step 1. In step 2, we

set the number of female spiders N_f to 5 and the number of male spiders N_m to 1. We initialize the iteration counter $t = 0$ in step 3. The initial population of female population fsp as shown in Equation 22.3 is created in steps 4–8 as follows.
$N1 = \{-5.3772, 2.1369, -0.2804, 7.8260, 5.2419\}$
$N2 = \{-0.8706, -9.6299, 6.4281, -1.1059, 2.3086\}$
$N3 = \{5.8387, 8.4363, 4.7641, -6.4747, -1.8859\}$
$N4 = \{8.7094, 8.3381, -1.7946, 7.8730, -8.8422\}$
$N5 = \{-2.9426, 6.2633, -9.8028, -7.2222, -5.9447\}$
while the population of male msp as shown in Equation 22.4 is generated in steps 9–13 as follows. $N6 = \{-6.0256, 2.0758, -4.5562, -6.0237, -9.6945\}$
Each solution in the population (females and males) is evaluated by calculating its weight (fitness function) as shown in Equation 22.5 in steps 15–17 as follows.
In step 16, the values of the objective function for each solution in the populations $J(.)$ are $\{j = 122.2832, 141.3675, 173.4364, 288.7659, 231.4821, 191.6448\}$. The values of the overall worst and best solutions in the population are $worsts = 288.7659$, $bests = 122.2832$, respectively. The quality for each solution is assigned by calculating the weight of each solution in the population as follows $w_i = \{1.0010, 0.8864, 0.6937, 0.0010, 0.3451, 0.5844\}$. In steps 18–19, we calculate the vibrations of the best local and global solutions $Vibc_i$ and $Vibb_i$, where $Vibc_i = 1.5753$ and $Vibb_i = 1.7244$. In step 20, the random number $rm = 0.1934$ is less than the threshold $PF = 0.9048$ so the female spider is moving as shown in Equation 22.11 as follows:
α, β are random vectors, where $\alpha = \{0.6822, 0.3028, 0.5417, 0.1509, 0.6979\}$, $\beta = \{0.3784, 0.8600, 0.8537, 0.5936, 0.4966\}$ and $\delta = 2 \times PF$, where $\delta = 1.8097$.
The new female solutions $f1 = \{-5.3772, 2.1369, -0.2804, 7.8260, 5.2419\}$
$f2 = \{-0.8706, -9.6299, 6.4281, -1.1059, 2.3086\}$
$f3 = \{5.8387, 8.4363, 4.7641, -6.4747, -1.8859\}$
$f4 = \{8.7094, 8.3381, -1.7946, 7.8730, -8.8422\}$
$f5 = \{-2.9426, 6.2633, -9.8028, -7.2222, -5.9447\}$.
Based on the previous values, the new female spiders' values are shown below
$f1 = \{-5.5953, 2.7373, -0.2753, 8.2051, 5.1133\}$
$f2 = \{7.8709, -32.3970, 22.2858, -11.8858, -2.9965\}$
$f3 = \{20.3480, 20.9845, 11.6847, -32.3357, -13.7400\}$
$f4 = \{23.3709, 16.5485, -3.0839, 7.9691, -12.0286\}$
$f5 = \{-1.3881, 11.5068, -24.9425, -21.5623, -11.1657\}$.
In step 26, the median male equals 0.5844. In steps 27–34, the male solutions are updated according to the male weight $w_{N_f i}$ value and the median weight w_{N_f+m}, where $w_{N_f i} = 0.3451$ and $w_{N_f+m} = 0.5844$. The random vector is α where $\alpha = \{0.9891, 0.0359, 0.3798, 1.3668, 0.8667\}$. The vibration of the male spider $Vibf_i = 1.0172$.
The nearest female solution is $s_f - m_i^t$, where

$$s_f - m_i^t = \{-1.6925, -9.6998, 5.7772, -5.7207, -3.5993\}.$$

According to the previous values, the new male spider m_i^{t+1} is $m_i^{t+1} = \{-2.1862, 2.2598, -0.3307, -6.2653, -4.8513\}$. In step 35, the radius of mating in Equation 22.13 is calculated and set to $r = 3.4991$. The surviving operator in lines 36-46 is starting to generate a new male solution as follows $spbest = \{-1.9968, 2.7438, -0.6281, 1.4694, 2.3207\}$. In step 47 the iteration counter is increased and the termination criteria is checked in step 48. Finally the best solution is produced, which is {-2.1862,2.2598,-0.3307,-6.2653,-4.8513} and its value = 72.7846. The overall processes are repeated until termination criteria are satisfied.

22.6 Conclusion

In this chapter, we show the main steps of the SSO algorithm and how it works. We demonstrate the source code in Matlab and C++ language in order to help the user to implement it on various applications. In order to give a more thorough understanding of the SSO algorithm, we present a step by step numerical example and show how it can solve a global optimization problem.

References

1. L. Aviles. “Sex-Ratio bias and possible group selection in the social spider snelosimus eximius”. The American Naturalist, vol. 128, no. 1, pp. 1–12, 1986.
2. L. Avilés. “Causes and consequences of cooperation and permanent-sociality in spiders”. In B. C. Choe, The Evolution of Social Behavior in Insects and Arachnids pp. 476-498, Cambridge, Massachusetts.: Cambridge University Press, 1997.
3. T.R. Chandran, A. V. Reddy, and B. Janet. “An effective implementation of Social Spider Optimization for text document clustering using single cluster approach”. In 2018 Second International Conference on Inventive Communication and Computational Technologies (ICICCT) (pp. 508-511). IEEE, 2018.
4. E. Cuevas, M. Cienfuegos, D. Zaldívar and M. Pérez-Cisneros. “A swarm optimization algorithm inspired in the behavior of the social-spider”. Expert Systems with Applications, vol. 40, no. 16, pp. 6374–6384, 2013.

5. J.Q. James and V.O. Li. “A social spider algorithm for global optimization”. Applied Soft Computing, vol. 30, pp. 614–627, 2015.
6. C. Eric and K. S. Yip. “Cooperative capture of large prey solves scaling challenge faced by spider societies”. Proceedings of the National Academy of Sciences of the United States of America, vol. 105, no. 33, pp. 11818–11822, 2008.
7. S. Maxence. “Social organization of the colonial spider Leucauge sp. in the Neotropics: vertical stratification within colonies”. The Journal of Arachnology, vol. 38, pp. 446–451, 2010.
8. M.A. Tawhid, A. F. Ali. “A simplex social spider algorithm for solving integer programming and minimax problems”. Memetic Computing, vol. 8, no. 3, pp. 169–188, 2016.
9. M.A. Tawhid, A. F. Ali. “A hybrid social spider optimization and genetic algorithm for minimizing molecular potential energy function”. Soft Computing, vol. 21, no. 21, pp. 6499–6514, 2017.
10. A. Pasquet. “Cooperation and prey capture efficiency in a social spider, Anelosimus eximius (Araneae, Theridiidae)”. Ethology, vol. 90, pp. 121–133, 1991.