UNIVERSIDAD NACIONAL DE COLOMBIA
SEDE MEDELLÍN
FACULTAD DE MINAS

DYNA

http://dyna.medellin.unal.edu.co/

Discrete Particle Swarm Optimization in the numerical solution of a system of linear Diophantine equations

Optimización por Enjambre de Partículas Discreto en la Solución Numérica de un Sistema de Ecuaciones Diofánticas Lineales

Iván Amaya [a], Luis Gómez [b] & Rodrigo Correa [c]

[a] *PhD(c), Universidad Industrial de Santander, Colombia. ivan.amaya2@correo.uis.edu.co*
[b] *BSc on Electronics Engineering, Physicist, Universidad Industrial de Santander, Colombia. luisgomezardila@gmail.com*
[c] *Professor, PhD School of Electric, Electronic and Telecommunication Engineerings, Universidad Industrial de Santander, Colombia. crcorrea@uis.edu.co*

Received: February 25th, 2013. Received in revised form: January 31th, 2014. Accepted: April 3th, 2014.

Abstract
This article proposes the use of a discrete version of the well known Particle Swarm Optimization, DPSO, a metaheuristic optimization algorithm for numerically solving a system of linear Diophantine equations. Likewise, the transformation of this type of problem (i.e. solving a system of equations) into an optimization one is also shown. The current algorithm is able to find all the integer roots in a given search domain, at least for the examples shown. Simple problems are used to show its efficacy. Moreover, aspects related to the processing time, as well as to the effect of increasing the population and the search space, are discussed. It was found that the strategy shown herein represents a good approach when dealing with systems that have more unknowns than equations, or when it becomes of considerable size, since a big search domain is required.

Keywords: Linear Diophantine equations; objective function; optimization; particle swarm.

Resumen
El presente artículo propone utilizar una versión discreta del bien conocido algoritmo metaheurístico de optimización por enjambre de partículas, DPSO, para solucionar numéricamente un sistema de ecuaciones Diofánticas lineales. Así mismo, se muestra la transformación de este tipo de problema (es decir, la solución de un sistema de ecuaciones), en uno de optimización. El presente algoritmo es capaz de encontrar todas las raíces enteras en un dominio de búsqueda dado, al menos para los ejemplos mostrados. Se utilizan algunos problemas sencillos para verificar su eficacia. Además, se muestran algunos aspectos relacionados con el tiempo de procesamiento, así como con el efecto de incrementar la población y el dominio de búsqueda. Se encontró que la estrategia mostrada aquí representa una propuesta adecuada para trabajar con sistemas que tienen más incógnitas que ecuaciones, o cuando se tiene un tamaño considerable, debido a que se requiere un gran dominio de búsqueda.

Palabras clave: Ecuaciones Diofánticas lineales; enjambre de partículas; función objetivo; optimización.

1. Introduction

With each passing day is easier to see the boom that the modeling and description of systems have generated in science and engineering, especially through Diophantine equations. Areas such as cryptography, integer factorization, number theory, algebraic geometry, control theory, data dependence on supercomputers, communications, and so on, are some examples [1]. Moreover, there is a strong mathematical foundation for this type of equations and their solutions (both, at a fundamental and at an applied level). These vary from the fanciest and most systematic approaches, up to the most recursive ones, but it is evident that there is no unified solution process, nor a single alternative for doing so. Furthermore, some equations may have a single solution, while others may have an infinite number, or, possibly, may not even have a solution in the integer or rational domains. This also applies for linear systems with this kind of equations (i.e. Diophantine ones) [2]. Matiyasevich, during the early 90s, proved that it was not possible to have an analytic algorithm that allows to foresee if a given Diophantine equation has, an integer solution , or not [3]. This problem may have been one of the engines that have boosted the search for numerical alternatives.

In order to solve a system of linear Diophantine equations, a variable elimination method (which is quite similar to Gauss's) is a good approach for small systems, but it becomes demanding for bigger ones. The specialized literature report some methods like those based on the theory of modules over main ideal domains, which are somewhat more systematic when looking for all the solutions of a given system, but, likewise, become too complex when dealing with big systems of equations [4], [5]. Some authors have previously proposed the solution of a Diophantine equation through artificial intelligence algorithms [6], [7]. This article proposes to solve, in case the solution exists in the given search domain, a linear system of Diophantine equations. Initially, some basic and necessary related concepts are laid out, and then the viability of using the numeric strategy is shown through some examples.

2. Fundamentals

A linear Diophantine equation, with n unknowns, is defined by eq. (1), where $a_1, a_2, \dots, a_n$ are known rational, or integer, numbers, and $x_1, x_2, \dots, x_n$ are unknowns, i.e., the numbers that should satisfy them, [8]; b is a known integer. It is said that the integers $t_1, \dots, t_n$ are a solution for eq. (1) if, and only if, $a_1t_1 + \cdots + a_nt_n = b$.

$$a_1x_1 + a_2x_2 + \cdots + a_nx_n = b \tag{1}$$

One of the basic results of number theory that can be applied to a linear Diophantine equation is the following theorem, which allows determining whether it has a solution or not (even if it is not able to calculate it):

Theorem 1. Let $a_1, \dots, a_n, b$ be integers, where all a_i not zeros, and let $d = g.c.d.\{a_1, \dots, a_n\}$ be the g.c.d. of the numbers $a_1, \dots, a_n$. Therefore, $d|b$ if, and only if, exist $t_1, \dots, t_n$ integers, such that $a_1t_1 + \cdots + a_nt_n = b$.

Thus, the problem of determining whether a linear Diophantine equation has a solution or not, is reduced to showing if the greatest common divisor of the a_i coefficients divide b or not. Consider the case of two unknowns, for example, with an equation as the one shown by eq. (2), where a, b, c are known integers, and whose solution only exists if the g.c.d. of a and b is a divisor of c.

$$a * x + b * y = c \tag{2}$$

According to the previously mentioned theorem, this equation has integer solutions, and it can be shown that if (x_0, y_0) is a particular one, then all its solutions are given by eq. (3), where β is an integer and d is an integer which represents the g.c.d.

$$\begin{aligned} x &= x_0 + \beta * \frac{b}{d} \\ y &= y_0 - \beta * \frac{a}{d} \end{aligned} \tag{3}$$

Therefore, if a linear Diophantine equation with two unknowns has a solution in the integers, then it has infinite solutions of this kind. Even so, the problem now transforms in finding a particular solution, which can be done using the following method.

Let X be a non-empty subset of $\mathbb{R}^n$ and consider eq. (4), where $f: X \to \mathbb{R}$ is a function.

$$f(x) = 0, \qquad x \in X \tag{4}$$

The problem of finding all the possible solutions for eq. (4) in the subset X can be transformed into a global optimization problem over X as follows:

Let $g: X \to \mathbb{R}$ be defined by:

$$g(x) := [f(x)]^2 \tag{5}$$

Then, for every $x \in X$ it holds that $g(x) \geq 0$.

Theorem 2. Suppose that eq. (4) has a solution in X, and let $a \in X$. Therefore, a is a solution for eq. (4) if, and only if, a minimizes the function g defined in (5).

An immediate consequence of the previous theorem is that if eq. (4) has a solution in X, then the global minimum of g defined in (5) exists and is zero; even more, the following theorem exists:

Theorem 3. If the function g defined in (5) has a global minimum in X and this value is zero, then eq. (4) has a solution in X. Moreover, all global minimizers of g are solutions of eq. (4).

Then, if for $f(x_1, x_2) = a_1x_1 + a_2x_2 - b$ a region of the plane can be determined, where a global minimum of function g, defined by (5), and its value is zero, then any global minimizer with integer coordinates, should it exist, serves as a particular solution of eq. (2). Thus, the choice of the region is quite important to enclose, at least, a solution with integer coordinates.

2.1. System of linear equations

Consider the following system of m linear Diophantine equations, with unknowns $x_1, \dots, x_n$.

$$\begin{cases} a_{11}x_1 + \cdots + a_{1n}x_n = b_1 \\ \qquad\vdots \\ a_{m1}x_1 + \cdots + a_{mn}x_n = b_m \end{cases} \tag{6}$$

According to theorem 1, in order for the system (6) to have a solution, it is necessary, but not sufficient, that each of the m equations have a solution; this is equivalent to establishing if for each $i = 1, \dots, m$ it holds that $g.c.d.\{a_{i1}, \dots, a_{in}\}$ divides b_i.

To see why this condition is not sufficient, consider the system of Diophantine equations defined by

$$\begin{cases} \mathrm{x} + 3\mathrm{y} = -1 \\ \mathrm{x} + \ \mathrm{y} = \ \ 4 \end{cases} \tag{7}$$

Each equation from this system has a solution in the integer domain, but the system does not have a solution as a whole. Then, and in the same way that with systems of equations in real variables, the fact that one of the equations of a system has a solution, does not imply that the whole system also has.

Even so, a method that generalizes finding all the roots (in case they exist) of a system of equations over a given set, is shown below.

Let X be a non-empty subset of $\mathbb{R}^n$ and consider the system of equations (8), where for each $i = 1, \dots, m$, $f_i: X \to \mathbb{R}$ is a function.

$$\begin{cases} f_i(x) = 0 \\ \vdots \\ f_m(x) = 0 \end{cases} \quad where\ x \in X \tag{8}$$

Let $g: X \to \mathbb{R}$ be defined by:

$$g(x) := \sum_{i=1}^{m} [f_i(x)]^2 \tag{9}$$

Then for all $x \in X$ it holds that $g(x) \geq 0$. The following result is achieved:

Theorem 4. Suppose that the system of equations (8) has a solution in X, and let $a \in X$. Then, a is a solution of the system (8) if, and only if, a minimizes the function g defined in (9).

The general condition of the theorem 4 about the feasibility of solving the system (8) is important, since it is possible that the function g defined in (9) can be globally minimized but that the system (8) does not have a solution.

An immediate consequence of theorem 4 is that if the system (8) has a solution in X, then the global minimum of g defined in (9) exists and it is zero; moreover, the following result exists:

Theorem 5. If the function g defined in (9) has a global minimum in X and this value is zero, then the system (8) has a solution in X. Moreover, all global minimizers of g are solutions of the system (8).

Therefore, for the function g defined in (9), if there does not exist a global minimum in X or if it exists but is different from zero, then the system of equations (8) does not have a solution in X.

A basic result of the mathematical analysis of the algorithm establishes that if X is a compact set (i.e. closed and bounded) and g is continuous over X then the global minimum exists. Now, for g to be continuous in X it is enough that each f_i is continuous in X.

For the case of systems of Diophantine equations, unlike the particular case of an equation with two unknowns, the fact that a solution exists does not imply that others do, and even less that an infinite number exists.

For the search of possible solutions of a system of Diophantine equations, it must hold that the set X have points with integer coordinates, i.e. that $X \cap \mathbb{Z}^n \neq \emptyset$.

2.2. The algorithm

The implemented algorithm is built up from various interconnected blocks and is similar to the structure of traditional PSO (for real numbers), [9], [10]. A first stage is given by the random assignation of a swarm of user defined integers. Any size can be used here. Likewise, the definition of these values is subject to previous knowledge of the objective function (fitness), as well as to the presence of restrictions. Moreover, an initial speed of zero can be defined for the particles. After that, the algorithm evaluates, in the given search space, the objective function. With it, local and global best values are established, and both, speed and position, of each particle, are reevaluated as shown below. This procedure is iterative and is repeated until the convergence criteria are met, or until all solutions in the search domain are found.

An algorithm, considered as a variant of the traditional PSO, was used, [9]. In the same fashion as said PSO, its version for discrete solutions includes two vectors $\mathrm{X_i}$ and $\mathrm{V_i}$, related to the position and speed of each particle, for every iteration. The first one is a vector of random numbers, initially, in a valid solution interval. The second one can also be a random vector, but it can be assumed as zero for the first iteration, in order to keep it simple. When the problems become multidimensional, the vectors transform into a position and a speed matrices, since there is a value for each unknown, [9], [11]. Discrete PSO differs from its traditional version in which the new speed and position depend on both, an equation and a decision rule, which chooses among the local and global best values for the next iteration. Assuming there is a vector $y_i = (y_{i1}, y_{i2}, \cdots, y_{1n})$ that allows the transition between continuous and discrete PSO, and which takes the value of (-1, 1, or, 0) according to eq. (10), where glo is the global optimum of the swarm, and loc the local one, [9].

$$y_i = \begin{cases} 1 & \text{if} & \mathrm{X_i} = glo \\ -1 & \text{if} & \mathrm{X_i} = loc \\ 0 & \text{if} & \mathrm{X_i} \neq glo \neq loc \\ -1\ or\ 1 & \text{if} & \mathrm{X_i} = glo = loc \end{cases} \tag{10}$$

Afterwards, speed is updated according to eq. (11), where w is known as the inertia factor, which is used to limit the speed of the particles; $\mathrm{c_1, c_2}$ are constants which is usually are considered as equal to two; and $\mathrm{r_1, r_2}$ are random numbers between zero and one [10].

$$\begin{aligned} \mathrm{V_{i+1}} &= \mathrm{V_i} * \mathrm{w} + \mathrm{c_1} * \mathrm{r_1} \\ &* (-1 - \mathrm{y_i}) + \mathrm{c_2} * \mathrm{r_2} \\ &* (1 - \mathrm{y_i}) \end{aligned} \tag{11}$$

Then, the decision parameter, vector $B_i = (B_{i1}, B_{i2}, \cdots, B_{in})$, is calculated according to eq. (12).

$$B_i = y_i + \mathrm{V_{i+1}} \tag{12}$$

This parameter decides if the next position of the particle is chosen as the local or global best, or if it is chosen as a random number in the search domain. Thus, position update is done according to eq. (13), where α is a constant that defines the intensification (new position equal to the local or global bests) and the diversification (new position equal to a random number) [9].

$$X_{i+1} = \begin{cases} glo & \text{if} \quad B_i > \alpha \\ loc & \text{if} \quad B_i < -\alpha \\ \text{Int Rand} & \text{if} \quad -\alpha \leq B_i \leq \alpha \end{cases} \tag{13}$$

3. Results and Analysis

This section shows the results achieved after solving some systems of linear Diophantine equations, as an example of the method. A computer with an AMD Turion X2 Dual Core RM-72 processor, at 2.1 GHz, and with 4 GB of RAM memory, was used. During all the examples, the following parameters were used: w = 0.75, c_1 = 0.8, c_2 = 0.2, y, $\alpha = 0.3$. These values were chosen based on some preliminary tests and on the information available in the literature [1], [9].

3.1. System of equations A

It is required to solve the system given by eq. (14), in the set of positive integers, which represents the amount of animals bought by a farmer and its cost. The full statement of the problem is as follows: *"A farmer spent 10.000.000 COP, on 100 animals: chickens (x), pigs (y) and cows (z). if he bought the chickens at 5.000 COP, pigs at 100.000 COP and cows at 500.000 COP, and if he acquired animals of all three classes, how many did he buy of each one?"* [12].

$$\begin{aligned} x + y + z &= 100 \\ x + 20y + 100z &= 2000 \end{aligned} \quad (14)$$

This system is equivalent, by Gaussian reduction, to the system $\begin{cases} x - \frac{80}{19}z = 0 \\ y + \frac{99}{19}z = 100 \end{cases}$

Its general solution is given by: $= \frac{80}{19}t$, y= $100 - \frac{99}{19}t$, $z = t$, which for $t = 19$ yields: x = 80; y = 1; z = 19. In order to solve this problem with the discrete PSO algorithm, the following objective function is created:

$$F = (x + y + z - 100)^2 + (x + 20y + 100z - 2000)^2 = 0$$

After 20 runs of the algorithm, with a swarm of 1000 particles, the same answer was always achieved. Their duration, however, varied from 1.186 s, with 204 iterations, and up to 474.043 s, with 66832 iterations. It can then be concluded that, for this system, the algorithm delivers an answer with excellent precision and accuracy, even though the number of iterations and the duration were variable. It was found that their relationship is quite close to linearity (R^2=0.9955).

3.2. System of equations B

Afterwards, the system of seven linear Diophantine equations shown by (15) was solved, which represents a closed-loop control system, with unitary feedback, and where it is required to find the controller ($C(S)$), with six poles at $S = -1$ for the plant G(s) $= \frac{S^2+S+1}{S^3+3S^2+4S+3}$.

$$X_1 - 1 = 0 \quad (15$$

)

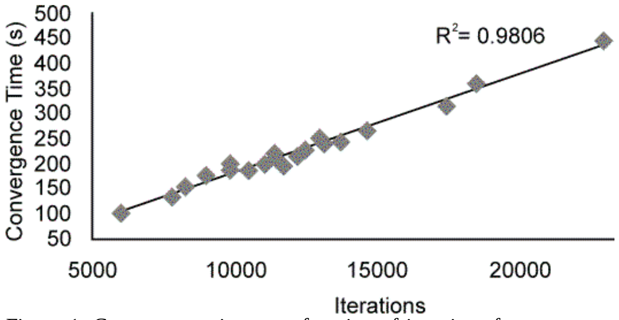

Figure 1. Convergence time as a function of iterations for system B.

$$\begin{gathered} 3X_1 + X_2 - 6 = 0 \\ 4X_1 + 3X_2 + X_3 + X_5 - 15 = 0 \\ 3X_1 + 4X_2 + 3X_3 + X_4 + X_5 + X_6 - 20 = 0 \\ 3X_2 + 4X_3 + 3X_4 + X_5 + X_6 + X_7 - 15 = 0 \\ 3X_3 + 4X_4 + X_6 + X_7 - 6 = 0 \\ 3X_4 + X_7 - 1 = 0 \end{gathered}$$

The solution of the system can be found to be:

$X_1 = 1, X_2 = 3, X_3 = 2, X_4 = 2, X_5 = 0, X_6 = -3,\ X_7 = -5$

From the first equation, $X_1 = 1$; and from the second one, $X_2 = 3$. The third equation yields $X_5 = 2 - X_3$, while from the fifth and sixth equations, $X_6 + X_7 = 6 - 3X_3 - 4X_4 = 6 - X_5 - 3X_4 - 4X_3$, which means that $X_4 = 2$. Thus, the last equation provides $X_7 = -5$. Substracting the fourth and fifth equations, $X_3 = 2$ is obtained, which means that $X_5 = 0$. Finally, the sixth equation yields $X_6 = -3$. In order to solve it through the algorithm, the following objective function was defined:

$$\begin{aligned} F = (X_1 - 1)^2 &+ (3X_1 + X_2 - 6)^2 \\ &+ (4X_1 + 3X_2 + X_3 + X_5 - 15)^2 \\ &+ (3X_1 + 4X_2 + 3X_3 + X_4 + X_5 + X_6 - 20)^2 \\ &+ (3X_2 + 4X_3 + 3X_4 + X_5 + X_6 + X_7 - 15)^2 \\ &+ (3X_3 + 4X_4 + X_6 + X_7 - 6)^2 \\ &+ (3X_4 + X_7 - 1)^2 = 0 \end{aligned}$$

Once again, 1000 particles were used and the algorithm was run 20 times. As a result, the same answer is achieved, so it is important to remark the excellent quality of the results (in terms of accuracy and precision), as well as, the variability in time and iterations, when looking for all the solutions in the integer domain. When compared to the previous system, it can be seen that the convergence time increased, and an almost linear relation between iterations and time can be seen in Fig. 1.

3.3. System of equations C

For this case a system of 12 linear Diophantine equations was selected:

$$\begin{aligned} 5x_1 - 6x_2 + 8x_4 - 5x_5 + 6x_6 + 10x_7 - 9x_9 + 3x_{10} + 11x_{11} \\ - 15x_{12} + 17x_{13} = -1 \end{aligned}$$

$$
\begin{aligned}
&7x_1 + x_2 - 4x_4 + 6x_7 - 9x_8 + 5x_9 - 12x_{10} + 3x_{11} - 7x_{12} \\
&\qquad + 8x_{13} = 26 \\
&5x_1 - 24x_2 + 32x_3 - 49x_4 + 3x_5 + 19x_6 - 21x_7 - 17x_8 \\
&\qquad + 33x_9 + 9x_{10} - 12x_{11} - x_{13} = 475 \\
&20x_1 + 27x_2 - 23x_4 - 30x_5 + 34x_6 + x_7 - 7x_9 + 11x_{10} \\
&\qquad - 28x_{11} + 4x_{12} - 36x_{13} = 103 \\
&5x_1 - 10x_3 + 2x_5 - 6x_7 - 13x_9 + 34x_{11} - 9x_{13} = -352 \\
&x_2 + 22x_4 - 26x_6 - 17x_8 + 19x_{10} - 4x_{12} = -84 \\
&30x_1 + 24x_2 - 55x_3 - 15x_4 - 25x_5 + 10x_6 + 40x_7 - 10x_8 \\
&\qquad + 8x_9 - 3x_{10} - 16x_{11} + 4x_{12} - 20x_{13} \\
&\qquad = 283
\end{aligned}
$$

$$
\begin{aligned}
&5x_1 - 13x_2 + 7x_4 + x_6 - 19x_7 + 19x_8 - 2x_9 + 6x_{10} + 5x_{11} \\
&\qquad - 26x_{12} = -468 \\
&x_1 + 28x_2 + 33x_3 - 100x_5 + 5x_6 + 13x_7 - x_8 - x_9 + 11x_{10} \\
&\qquad - 7x_{11} - 3x_{12} + x_{13} = -100 \\
&7x_3 - 21x_4 + 35x_5 - 42x_6 + 7x_7 + 14x_8 - 35x_9 + 28x_{10} \\
&\qquad - 7x_{11} + 14x_{12} + 56x_{13} = 329 \\
&5x_7 + 5x_8 + 10x_9 - 50x_{10} + 20x_{11} - 25x_{12} + 30x_{13} = -345 \\
&2x_1 - 4x_2 + 4x_3 - 2x_4 - 6x_5 + 8x_6 + 10x_7 + 9x_8 - 12x_9 \\
&\qquad + 20x_{10} + 6x_{11} - 30x_{12} + 16x_{13} = -78
\end{aligned}
$$

whose solution is:

$x_1 = 1;\ x_2 = -3; x_2 = 2;\ x_4 = -1;\ x_5 = 3;\ x_6 = 7;\ x_7 = 9;\ x_8 = -4;\ x_9 = 5;\ x_{10} = 5;\ x_{11} = -5;\ x_{12} = 10;\ x_{13} = 6$

The objective function is, once again, built using the squared sum of each equation. A search space between -10 and 10 was defined, and 100 particles were used. On the same computer, an excellent quality answer (in terms of accuracy and precision) was found, but it required an average time of 129632 s (around 36 hours) and 1026435 iterations. It is worth mentioning that it was not possible to find these roots by using commercial software nor through traditional means. Fig. 2 shows the exponential increment in time, when expanding the search domain.

3.4. System of equations D

In order to further test the algorithm's effectiveness, some other Diophantine systems were used. However, in this case they do not have a solution in the set of integers, e.g. the system given by eq. (16), which has a range A = range (A ,C) = 2, and a g.c.d. (2,1,3) = g.c.d (8,−5,−3) = 1| { 7,11}.

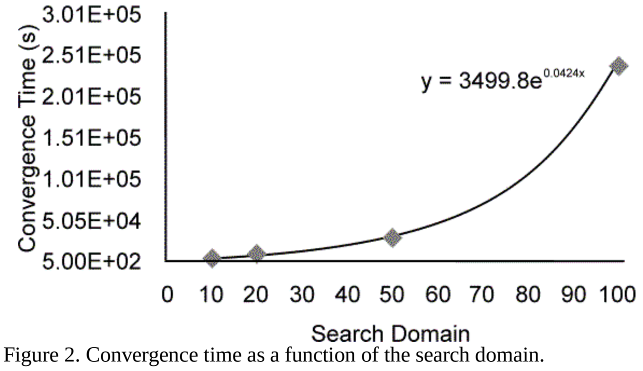

Figure 2. Convergence time as a function of the search domain.

$$
\begin{aligned}
2x + y + 3z &= 7 \\
8x - 5y + 3z &= 11
\end{aligned}
\tag{16}
$$

The discrete PSO algorithm reports that after 20 or more runs, for different swarm sizes and parameters, it was not possible to find an answer. It was also observed that if a system, e.g. the one given by eq. (17), has infinite solutions, a search domain must be defined, striving to locate solutions over this given set.

$$
\begin{aligned}
10w + 3x + 3y + 8z &= 1 \\
6w - 7x - 5z &= 2
\end{aligned}
\tag{17}
$$

4. Conclusions

This research proved that it is possible to numerically solve a system of linear Diophantine equations through an optimization algorithm. Also, it was observed that it is possible to solve this optimization problem without using conventional approaches. It was shown, through some simple examples, that, at least for these systems, solutions with high precision and accuracy are achieved. Moreover, it was found that the convergence time and the number of iterations are random variables that mainly depend on factors such as the algorithm parameters, the initial swarm and the size of the system. Obviously, when solving a squared, small system, traditional approaches, including the ones found in most of the commercial mathematical software, are far quicker, even those that find all the roots of the system. However, in case that it is required to solve a system with more unknowns than equations, a typical situation, they are out of the question. Likewise, if the system is of a considerable size, the convergence time drastically increases, since a big search domain is required (a case found during the current research), so the numerical strategy proposed here gains importance as a possible solution alternative.

References

[1] Abraham, S., Sanyal, S., and Sanglikar, M., Particle Swarm Optimization Based Diophantine Equation Solver, ArXiv, pp.1–15, Mar. 2010.

[2] Bonilla E., M., Figueroa G., M., and Malabare, M., Solving the Diophantine Equation by State Space Inversion Techniques : An Illustrative Example, Proceedings of the 2006 American Control Conference, pp.3731–3736, 2006.

[3] Matiyasevich, Y. V., Hilbert’s Tenth Problem. MIT Press, , 1993.

[4] Wu - Shr-Hua. Time-varying feedback systems design via Diophantine equation order reduction, thesis (Ph.D. on Electrical Engineering), United States, The University of Texas at Arlington, 2007, pp. 1–140.

[5] Cohen, H., Number Theory, Vol. I: Tools and Diophantine Equations and Vol. II: Analytic and Modern Tools. Springer-Verlag, New York, 2007.

[6] Lugar, G., Artificial Intelligence: Structures and Strategies for Complex Problem Solving. Addison-Wesley, Boston, 2006.

[7] Abraham, S. and Sanglikar, M., Finding Numerical Solution to a Diophantine Equation: Simulated Annealing as a Viable Search Strategy, Proceedings of the International Conference on Mathematical Sciences, 2, pp.703–712, 2008.

[8] Contejean, E., An Efficient Incremental Algorithm for Solving Systems of Linear Diophantine Equations, Information and Computation, (113), pp.143–172, 1994.

[9] Jarboui, B., Damak, N., Siarry, P., and Rebai, A, A combinatorial particle swarm optimization for solving multi-mode resource-constrained project scheduling problems, Applied Mathematics and Computation, 195 (1), pp.299–308, Jan. 2008.

[10] Amaya, I., Cruz, J., and Correa, R., Real Roots of Nonlinear Systems of Equations Through a Metaheuristic Algorithm, Revista Dyna, 78 (170), pp.15–23, 2011.

[11] Amaya, I., Cruz, J., and Correa, R., Solution of the Mathematical Model of a Nonlinear Direct Current Circuit Using Particle Swarm Optimization, Revista Dyna, 79 (172), pp.77–84, 2012.

[12] Gonzáles - F. J. Ecuaciones Diofánticas, thesis (Apuntes de Matemática Discreta), Universidad de Cádiz, 2004, pp. 353–354.

I. Amaya, received his bachelor degree on Mechatronics Engineering from Universidad Autónoma de Bucaramanga, Bucaramanga, Santander (Colombia). Currently, he is with the School of Electrical, Electronic and Telecommunications Engineerings and is pursuing his PhD on Engineering at Universidad Industrial de Santander, Bucaramanga, Santander (Colombia). His research interests include global optimization and microwaves.
ORCID: 0000-0002-8821-7137

L. Gómez, received his bachelor degree on Physics from Universidad Industrial de Santander Bucaramanga, Santander (Colombia), and also a bachelor degree on Electronics Engineering from the same University. His research interests include global optimization and Diophantine equations.

R. Correa, received his bachelor degree on Chemical Engineering from Universidad Nacional de Colombia, Bogotá, Cundinamarca (Colombia), and his master degree on Chemical Engineering from Lehigh University, Bethlehem, Pensilvania (USA) and from Universidad Industrial de Santander, Bucaramanga, Santander (Colombia). He received his PhD from Lehigh University on Polymer Science and Engineering and is currently a professor at Universidad Industrial de Santander. His research interests include microwave heating, global optimisation, heat transfer and polymers.