Diophantine Equations and Fuzzy Adaptive Simulated Annealing

HIME AGUIAR e O. Jr.
COPPE/UFRJ
Program of Electrical Engineering
Rio de Janeiro
BRAZIL
hime@engineer.com

Abstract: This work uses the Fuzzy Adaptive Simulated Annealing algorithm to solve many examples of Diophantine equations, offering a global optimization alternative for those ones that did not have a general method or even a closed form for their solution. The proposed method uses the built-in integer programming capabilities of Fuzzy ASA and the candidate solutions evolve in $\mathbb{Z}^n$, in contrast to the best known applications of ASA, exposing an almost unexplored feature of that paradigm. Some significant numerical results are introduced, and empirical relationships between number of independent variables and necessary number of cost function evaluations to get to solutions are presented.

Key–Words: Global optimization, Equation solving, Diophantine equations, Adaptive Simulated Annealing

1 Introduction

A Diophantine equation [3, 4] is a polynomial equation, whose general format is

$$f(x_1, x_2, ..., x_n; a_1, a_2, ..., a_n) = N \tag{1}$$

where the a_i and N are integer parameters and the x_i are the independent variables (they should also be integers).

In this fashion, solving equation (1) is equivalent to finding an integer n-tuple $(x_1^*, x_2^*, ..., x_n^*)$ that satisfies that functional relationship.

Diophantine equations got their name from mathematician Diophantus, who studied these equations many centuries ago. Those kind of problems are now referred to as Diophantine equations. Since these equations and their particular cases have always been of great interest to mathematicians and more recently have found many practical applications, research continues aimed at finding new methods for solving them in greater generality [1]. In the theoretical dimension, there is interest in finding the complete set of solutions for a given equation, as well as its cardinality and whether there is a deterministic procedure to find it, among several other, almost infinite issues. An important line of investigation is focused on elliptic curves, that can be used to help in designing public key cryptosystems. Elliptic curves are related to solutions of Diophantine equations with the following format

$$y^2 = x^3 + ax + b \tag{2}$$

where a and b are rational numbers and the right hand side has distinct roots.

Another very important type of equation is

$$x_1^n + x_2^n = x_3^n \tag{3}$$

that is definitely linked to the name of French mathematician Pierre de Fermat, who stated perhaps its most important property, known as the Fermat's last theorem, which took more than 300 years to be proved.

Even though there are a large number of types of Diophantine equations, in general their resolutions are not trivial and depend on many factors. This is so due to the fact that a Diophantine equation can be unsolvable or present a finite or infinite number of solutions. Even the great mathematician David Hilbert manifested interest in Diophantine equations and their solutions, and one of his famous problems is related to this type of equation. In this fashion, considering that there are no general methods for finding

solutions to them, new methods capable of synthesizing numerical solutions to Diophantine equations are welcome and very important in present days. On the other hand, this is a difficult problem, mainly due to the fact that the number of possible solutions can be enormous and the associated computational complexity will be proportionally high as well.

Once we have a Diophantine equation problem, the first thing to do is to try to find out a local solution for it, using one of the methods present in the literature [3]. If there is none, the problem is solved. Of course, the most interesting problems are those for which the equations are everywhere locally solvable and the local knowledge that we obtain may already completely solve the problem, or might furnish useful information on the global problem through local–global principles (generally speaking, a local–global principle is a statement that asserts that a certain property is true globally – usually in a number field – if and only if it is true everywhere locally – usually in p-adic fields. It is important to know when this is indeed true, and when it may not be true). However, such a situation is rather rare [3] and it may be imperative to make a global analysis of the equation, either by factoring over $\mathbb{Z}$ or by working in some appropriate number field K. There are many methods for doing this [3, 4]. The most classical method, originating with Fermat, Euler, Gauss and Kummer, applies when it is possible to factor the Diophantine equation in K, a typical example being Fermats last theorem, where one factors the equation in a specific cyclotomic field K. In this case, it is very important to know explicitly the structure of the class group of K and of the unit group, and to be able to find explicitly a generator of a principal ideal. It is worth to be aware that, nowadays, these types of problems can be solved using computer programs specifically designed for such tasks. Another global method for solving Diophantine equations is based on Diophantine approximation techniques and on Baker–type results on linear forms in logarithms of algebraic numbers [3, 11]. It is used in particular to solve Thue equations, that is to say, equations of the form f(x, y) = m, where f is a homogeneous polynomial in two variables. One more global method for solving certain kinds of Diophantine equations, mainly those that can be reduced to a cubic one, is based on the use of elliptic curves, be it via the method of infinite descent (initiated by Fermat, and which does not necessarily involve elliptic curves explicitly) or via the Birch and Swinnerton–Dyer conjecture. This important conjecture enables us to predict the $\mathbb{Z}$-rank of the group of points of an elliptic curve over $\mathbb{Q}$, by computing a purely analytic quantity and tells us whether this group is finite or infinite. The fact that this method is based on a conjecture is not significant, since either the analytic result states that the group is finite and, in that case, Birch and Swinnerton–Dyer conjecture is proved, or it concludes that the group is infinite, and we can obtain generators of the group by means of other techniques. A very sophisticated method for solving Diophantine equations is that used by Ribet, Wiles, and Taylor-Wiles for solving completely Fermat's last theorem, by means of modular forms and Galois representations. Usually, it is able to solve Diophantine equations of the form $a+b+c=0$, where a, b and c are highly divisible by certain integers. This method is based on a combination of a theorem of Ribet on "level lowering" of modular forms with the theorem of Wiles and Taylor–Wiles saying that the L–function attached to an elliptic curve defined over $\mathbb{Q}$ is in fact the L-function of a modular form. The proof of these theorems is not easy and Wiles's theorem was considered one of the greatest mathematical findings of the end of the twentieth century [3, 4, 11].

Going in this direction, this work applies the Fuzzy Adaptive Simulated Annealing global optimization method (Fuzzy ASA, for short) in order to find numerical solutions for a certain subset of this type of equations. Fuzzy ASA is a fuzzy controlled version of ASA, that uses a Cauchy like generating function with wider tails, expanding or contracting according to the dynamical variation of the cost function at each dimension of its domain. This scheme is an improvement of previous SA approaches and can be, in some cases, superior to evolutionary algorithms like genetic algorithms, particle swarm optimization or differential evolution [7, 8]. So, our aim in this article is to discuss the procedure able to approximate solutions to equations of type (4), shown below. Naturally, other class of equations can also be solved in the same way and that would be a very interesting challenge to face in the future. In the past there were some attempts to apply soft computing techniques to find numerical solution of Diophantine equations. For example, in ref. [1] the authors tried to find numerical solutions to Diophantine equations

by applying genetic algorithms to them, and the same was done in [2], this time using the particle swarm optimization paradigm. In all works presented in the literature, some kind of progress was done and additional research in that direction might be beneficial in terms of getting more effective methods in this area. However, we can observe that, in most works, very specific algorithms are used, being difficult to find very successful applications of unmodified general purpose algorithms to those problems. So, in what follows, we are going to apply the general purpose Fuzzy ASA method (not customized) to the same problem and assess its effectiveness.

As said before, in this paper we discuss a generic algorithm to solve the class of Diophantine equations given by

$$a_1 x_1^{p_1} + a_2 x_2^{p_2} + ... + a_n x_n^{p_n} = N \qquad (4)$$

The rest of the text is organized as follows: Section 2 describes the proposed algorithm. Section 3 explains the (Fuzzy) ASA paradigm and presents some of its many features and possibilities. Section 4 presents the experimental results and section 5 concludes with some observations about the findings exposed along the text.

2 Proposed algorithm

To find solutions for equation (4) we propose the following algorithm:

- 1 – Transform the original problem into a global optimization one, by taking as the objective function $F(\mathbf{x})$ the absolute value of $a_1 x_1^{p_1} + a_2 x_2^{p_2} + ... + a_n x_n^{p_n} - N$

$$F(\mathbf{x}) \stackrel{\Delta}{=} |a_1 x_1^{p_1} + a_2 x_2^{p_2} + ... + a_n x_n^{p_n} - N| \qquad (5)$$

- 2 – Submit $F(\mathbf{x})$ to the Fuzzy ASA algorithm (in its integer programming mode), aiming to find different solutions to the new global optimization problem defined by (5). Here, it is possible to simply run the algorithm several times or use its multistart version, taking into account that, as a stochastic method, Fuzzy ASA is able to explore different regions during different activations.

- 3 – If sufficient solutions were found, stop. Otherwise, repeat step 2 until satisfaction or meeting a certain stopping criterion (for instance, reaching a maximum number of trials).

The reasoning underlying the algorithm is extremely simple: if it is possible to minimize $F(\mathbf{x})$ and the minimum value is 0, then the original equation (4) is automatically satisfied. So, the quality of the algorithm is strongly based upon the minimization power of the employed global optimization algorithm (Fuzzy ASA, in the present case), particularly in its ability to face arbitrarily complex landscapes of highly nonlinear cost functions.

3 The Adaptive Simulated Annealing approach

As its name indicates, the ASA method [5, 6, 9, 10] is based upon the simulated annealing concept, but presenting a number of additional features. Here, we highlight just a few of them:

- Re-annealing – it is the dynamical re-scaling of parametric temperatures, adapting generating probability distribution functions to each dimension according to different sensitivities. In a few words, if the objective function does not present significative variations whenever a given parameter is altered, it could be advantageous to extend the search interval amplitude in that dimension in particular, and vice-versa.

- Quenching – the ASA implementation offers the possibility of adjusting several structural parameters related to the quenching process, allowing the user or any automatic control mechanism to change the default behavior of the "cooling" process and drive the evolution of the parametric temperatures. This device is useful in case of stagnation near suboptimal regions.

- High flexibility degree – the ASA system was idealized so as to allow the users to alter virtually any subsystem without significative programming effort. This way, it is possible to change the behavior of generation/acceptance processes, termination criteria, seed generation etc..

In practical cases, functions to be minimized show themselves in the form of cost measures that vary with several parameters and are subject to certain constraints, imposed by their environments. Whenever the objective function is well-behaved, that is, differentiable, convex or satisfying Lipschitz conditions, there are several methods capable of finding points at which it attains its minimum value, obeying certain imposed constraints. Difficulties arise whenever the given function presents several local minima, thus making the final result dependent on the starting point. Unfortunately, most real problems lead to very complex objective functions that are nonlinear, discontinuous, multi-modal and multi-dimensional among other properties. To solve such a class of problems, stochastic methods seem to be a good, and sometimes the only, alternative. Genetic algorithms (GA), simulated annealing (SA) and particle swarm optimization (PSO) are among the most popular approaches to stochastic global optimization. The difficulty associated to stochastic approaches is mainly related to the speed of convergence [13] but, in spite of these drawbacks, researchers have found ways to overcome certain limitations of original evolutionary schemes, leading to implementations such as adaptive simulated annealing (ASA) [5], which is a sophisticated and rather effective global optimization method. The ASA technique is particularly well suited to applications involving neuro-fuzzy systems and neural network training, thanks to its superior performance and simplicity. The ASA approach has the benefits of being publicly available, parameterized, and well-maintained, thereby showing an alternative to GA, according to the published benchmarks, which demonstrate its effectiveness [5]. Unfortunately, many stochastic global optimization algorithms share a few problems, such as large periods of poor improvement in their way to a global optimum. In SA implementations, that behavior is mainly due to the cooling schedule, whose speed is limited by the characteristics of probability density functions, which are employed with the purpose of generating new candidate points. In this manner, if we choose to employ the so called Boltzmann annealing, the temperature has to be lowered at a maximum rate of T(k) = T(0)/ln(k). In the case of fast annealing, the schedule becomes T(k) = T(0)/k, if assurance of convergence with probability 1 is to be maintained, resulting in a faster schedule. The approach based on ASA has an even better default scheme, because of its improved generating distribution.

ASA was designed to find global extreme inside a pre-established hyper-rectangle and generates points componentwise, according to the following scheme:

$x_{i+1} = x_i + \Delta x_i$, with $\Delta x_i = y_i(B_i - A_i)$,
$[A_i, B_i]$ = interval corresponding to i-th dimension,
$y_i \in [-1,1]$ is given by $y_i = sign(u_i - 1/2)T_i[(1 + 1/T_i)^{|2u_i-1|} - 1]$, where $u_i \in [0,1]$ is generated by means of the uniform distribution,
T_i = present temperature relative to dimension i.

The compactness of search space is not a severe limitation in practice and, in the absence of previous information about global optima location, it suffices to choose sufficiently abrangent hyper-rectangular domains.

As said before, the quenching mechanism can improve, in many cases, the efficiency of the convergence process, although there is always the risk of reaching prematurely nonglobal extreme. In certain scenarios, however, we might simply not have alternative ways out of a stagnation situation, as occurs with functions operating in high-dimensional spaces. Trying to overcome this difficulty, a fuzzy controller was designed (Fuzzy ASA) [6]. The approach is simple: the original ASA system is seen as a MISO (Multiple Input Single Output) dynamical system and the supplementary code simply ”closes the loop”, by sampling its output (current value of objective function) and acting in its inputs (a subset of run-time adjustable parameters, related to the quenching process) according to a fuzzy law (control algorithm), imitating human behavior whenever subject to similar underlying situations. In this way, active run-time control can accelerate or slow down temperature evolution, besides being able to take evasive actions, in case of premature convergence. In its present version, Fuzzy ASA code rises the quenching degree after detecting decreasing optimization performance or potential stagnation states, aiming to recover from an possible undesirable convergence to nonglobal optima. Please, note that this additional module does not try to substitute the many effective devices already present in the ”pure” ASA code, just complementing them in nontypical situations.

3.1 Additional considerations about reannealing

Whenever doing a multi-dimensional search in order to find a the solution for a nonlinear problem, we usually have to cope with different changing sensitivities of the coordinates of generated points (let us call them α^i) during the search. At any given annealing time, it seems sensible to attempt to "stretch out" the range over which the relatively insensitive parameters are being searched, in relation to the ranges of more sensitive parameters.

This may be done by periodically rescaling the annealing time k, that is to say, reannealing, every fixed number of acceptance events in terms of the sensitivities s_i, calculated at the most recently detected minimum value of the cost function, L,

$$s_i = \partial L / \partial \alpha^i. \tag{6}$$

In terms of the largest $s_i = s_{\max}$, ASA can reanneal by using a rescaling for each k_i of each parameter dimension,

$$k_i \to k'_i, \tag{7}$$

$$T'_{ik'} = T_{ik}(s_{\max}/s_i), \tag{8}$$

$$k'_i = \left(\ln(T_{i0}/T_{ik'})/c_i\right)^D. \tag{9}$$

T_{i0} is set to 1 to start the process, and that is sufficient to sweep all the amplitude of a given parameter.

The acceptance temperature is recalled in the same way. Besides, considering that the initial acceptance temperature is equal to a trial value of L, this is typically very large relatively to the global minimum. Therefore, when this rescaling is realized, the initial acceptance temperature is reset to the most current minimum of L, and the annealing time associated with this temperature is set to give a new temperature equal to the lowest value of the cost function found until present annealing time.

The standard deviations of the theoretical forms are evaluated as well, and calculated by $[\partial^2 L/(\partial \alpha^i)^2]^{-1/2}$, for each parameter α_i. This gives an estimate of the fluctuations that accompanies fits to stochastic data or functions. At the end of the run, the off-diagonal elements of the covariance matrix are calculated for all parameters. This inverse curvature of the theoretical cost function can provide an estimate of the relative sensitivity of parameters to statistical errors during minimization processes.

More devices like that can be added, but the truth is that nonlinear systems typically show a great deal of diversity and each one requires some observation in order to develop a truly efficient parameter configuration. Another feature of ASA is its ability to recursively self optimize its Program Options, for example the c_i parameters described above.

3.2 Quenching

A significant feature of ASA is its ability to perform quenching, and this mechanism can be applied by noting that the temperature schedule above may be redefined as

$$T_i(k_i) = T_{0i}\exp(-c_i k_i^{Q_i/D}), \tag{10}$$

$$c_i = m_i \exp(-n_i Q_i/D), \tag{11}$$

in terms of the quenching factor Q_i. The proof of convergence presented in [5] fails if $Q_i > 1$, considering that

$$\sum_k \prod^D 1/k^{Q_i/D} = \sum_k 1/k^{Q_i} < \infty. \tag{12}$$

This fact shows how the so called "curse of dimensionality" arises, and at the same time gives a possible way of coexist with it. In ASA, the influence of large dimensions becomes clear because the exponential of the power of k is $1/D$, so the annealing required to properly sample the space becomes too slow. In this way, if we don't have resources enough to properly sample the space ergodically, then the next best procedure could be to turn on quenching, whereby Q_i can become on the order of the number of dimensions.

The scale of the power of $1/D$ temperature schedule (used for the acceptance function) can be similarly altered. However, this does not affect the annealing proof of ASA, and so this may be used without damaging the weak ergodicity property [7].

3.3 VFSR and ASA

ASA is the natural evolution of the method called very fast simulated reannealing (VFSR) [5], named in this way to distinguish it from the previous method of fast annealing (FA) [12]. The annealing schedules for the temperatures T_i decrease exponentially in annealing time k, that is, $T_i = T_{0i}\exp(-c_i k^{1/D})$. Of

course, the fatter the tail of the generating function, the smaller the ratio of acceptance to generated points in the optimization run. But, when properly tuned, it is possible to see that for a given generating function this ratio is approximately constant as the minimization run finds a global minimum. Consequently, for a large parameter space, the efficiency of the minimization process is determined by the annealing schedule of the generating function.

A major difference between ASA and Boltzmann annealing algorithms is that the ergodic sampling takes place in an $(n+1)$–dimensional space $(\mathbb{R}^{n+1})$, that is, in terms of n parameters and the cost function. In ASA the exponential annealing schedules allow resources to be spent adaptively on reannealing and on speeding the convergence in all dimensions, ensuring full global searching in the first phases of search and quick convergence in the final phases. The chosen acceptance function presents the usual Boltzmann form, satisfying detailed balance, and the acceptance temperature reannealing conducts the convergence of the cost function to permit ergodic searching in the n-parameter space considered as the independent variables of the objective function.

3.4 Considerations about tuning ASA

As the ASA implementation offers the user more than 100 adjustable parameters in order to customize its performance, it is important to get some experience in setting them for a particular problem. After all, nonlinear systems are, typically, atypical. So, trying to fully understand the nature of the cost function under sampling in order to tune ASA by examining just the cost function, probably will not be so productive as generating more intermediate output, for instance, by setting ASA_PRINT_MORE to TRUE, and looking at this output as a "grey box" of insight into your optimization problem. Larger files with more information are provided by setting ASA_PIPE_FILE to TRUE. Treat the output of ASA as a simulation in the ASA parameter space, which usually is quite a different space than the variable space of your problem.

In some circunstances, you should be able to see where and how your solution might have been "captured" in a local minima for a very long period, or where the last saved state is still fluctuating across a large part of your state space. These observations should suggest how you might try speeding up or slowing down annealing/quenching of the parameter space and/or tightening or loosening the acceptance criteria at different stages by modifying specific parameters of ASA. The distribution comes with the ASA code and additional documentation, providing guidelines for tuning that may provide the user with some helpful insights. An especially important measure is to examine the output of ASA at several stages of sampling, to discover whether changes in parameter and temperatures are reasonably correlated to variations in the cost function.

Although it is good having many parameters available for tuning, the natural consequence is that it can take a while to get comfortable adjusting ASA parameters, especially if the default settings do not work well for your specific problem. In this case, it is advisable to examine ASA documentation and try to identify which parameters could be adequate to solve the particular problem.

Tuning is a necessary feature of any sampling algorithm if it is to be applied to many classes of systems. It is not adequate to compare stochastic global optimization algorithms that don't have proper mechanisms allowing us to tune each one to cost functions being optimized.

At present, the ASA implementation reflects the feedback given by thousands of users [7]. As said above, a major feature of the code is that it has more than a hundred tuning parameters, whereas in many other similar algorithms only a few are usually offered. Other very good sampling–based algorithms do not give such ample level of tuning, and too often do not work on some more complex functions exactly for this reason. In general, the more optional features you use, the more "weight" is added to the executable code but, since most of these ASA OPTIONS are chosen at pre–compile time, this does not affect the execution performance in runtime. According to [7] there were previous efforts to rewrite the code into another language, for instance, C++, Java, Matlab, etc., but they were abandoned due to difficulties in integrating all the ASA OPTIONS in the same way that they are presently implemented. The fact is that all ASA adjustable parameters (usually referred to as OPTIONS by ASA's creator) are really welcome, because they allow the users to tailor the same code to many different applications.

As said before, nonlinear systems are, typically,

atypical, and it is difficult to furnish guidelines for ASA default values of parameters, similarly to what is expected for ”off–the–shelf” quasi–linear systems. There is an ongoing effort to prepare some guidelines and convey them to the users by means of the ASA–README file, and also a special feature (SELF_OPTIMIZE parameter) that could help users to automatically tune parameters based on the behavior of specific cost functions under minimization. Probably the best approach would be a hybrid approach, making initial guesses and observing the resulting performance. If the final result is not satisfactory, use the SELF_OPTIMIZE mode - this process can take a while, because ASA truly samples the configuration space. When SELF_OPTIMIZE is turned on, for each call of the top-level ASA parameters selected, the ”inner” shell of your system's parameters are optimized, and this is performed for an optimization of the ”outer” top–level shell of ASA parameters. If, even after having taken this set of measures, you feel that the something needs to be improved, it is possible to turn quenching on.

Note that the probabilistic convergence of ASA to a global optimal point can be assured only in terms of sufficient conditions [5]. This being even quite a strong statement, since we can only importance–sample a large space in a limited period of time. So, if the objective function under minimization is not too pathological, it might be advisable to slow down the annealing mechanism in order to allow ASA to spend more time at each step for the sake of investigating the finer scales. This could be required if the problem looks different at different scales, for then you can often get trapped in local optima, and thus ASA could fail just as any other “greedy” quasi-Newton algorithm.

ASA has demonstrated many times that it is more efficient and gets to global minimizers better than other importance–sampling techniques [5], but this normally requires tuning some parameters. For example, and as cited in the documentation that goes along with the package, a quasi-Newton algorithm could be more efficient than ASA for a parabolic system.

As the default probability distribution used in ASA is fat–tailed and the effective widths of the parameters being searched change quite slowly with decreasing parameter temperatures, the trade–off is that the parameter temperatures may decrease exponentially and still evolving according to the sampling proof. In this fashion, ASA tends to find global minima when other sampling techniques might fail.

Besides, given the independence of cost and parameter temperature evolution, additional tuning of ASA is possible in many difficult minimization tasks. While the decreasing parameter temperatures change the way the parameter states are generated, the decreasing cost temperature changes the way the generated states are accepted. The sensitivity of the acceptance criteria to the cost temperature schedule can be very influential in many problems. Analyzing a few execution instances by using parameter ASA_PRINT_MORE set to TRUE can detect events of stagnation in local minima or not enough holding time, for example, and that would require tuning of some related parameters.

4 Implementation and experiments

To validate the proposed algorithm, we have coded each objective function in a separate dynamic link library that can be called by the generic optimization code (ASA/Fuzzy ASA). This way, each specific problem can be encapsulated in one self-contained module.

To compare our findings to previous, recent research, we followed the line of testing adopted in reference [2]. In this fashion, several equations were submitted to the proposed algorithm and the results were tabulated. The most significant cases involved equations of the general form

$$x_1^2 + x_2^2 + ... + x_n^2 = N \tag{13}$$

and

$$x_1^n + x_2^n = N \tag{14}$$

where the integer n is variable and N can assume any positive integer value, allowing us to estimate, among other facts, the relationship between the dimension of the search space and the number of cost function evaluations necessary to find a solution (in the case of family (13)).

The original fuzzy ASA code and the same ASA parameters were kept intact through all tests in order to assess ASA's minimization ability without exploiting specific features of objective functions. The most significant ones are listed below.

$$\text{TemperatureRatioScale} = 1\text{E}-5$$
$$\text{CostParameterScaleRatio} = 1.0$$
$$\text{TemperatureAnnealScale} = .01$$
$$\text{UserInitialParameters} = \text{TRUE}$$
$$\text{InitialParameterTemperature} = 1000.0$$
$$\text{ReannealCost} = 1$$
$$\text{ReannealParameters} = \text{TRUE}$$
$$\text{IncludeIntegerParameters} = \text{TRUE}$$

It is also essential to configure ASA code so that independent variables be generated as integers (symbolic constant INTEGER_TYPE). In the sequel we will present the results corresponding to (13) and (14). Each particular test was run 30 times and the values relative to cost function evaluations correspond to averages taken over the respective samples. All tests were carried out using an Intel Core(TM) 2 CPU @ 2.4 Ghz with 512 Mb of RAM, and the programs were coded in the C++ programming language (Borland C++ compiler). It is worth to alert the reader that, owing to differences between implementations of numerical runtime libraries corresponding to the several programming environments available at present, it is possible for a particular numerical solution to produce slightly different functional results, when submitted to distinct computer programs, coded in different languages.

4.1 Equations of the type $x_1^2 + x_2^2 + ... + x_n^2 = N$

The following equations were included in the experiments

$$\sum_{i=1}^{12} x_i^2 = 3842\ ,\ \sum_{i=1}^{13} x_i^2 = 13000\ ,\ \sum_{i=1}^{14} x_i^2 = 14000$$
$$\sum_{i=1}^{15} x_i^2 = 15000\ ,\ \sum_{i=1}^{16} x_i^2 = 16000\ ,\ \sum_{i=1}^{17} x_i^2 = 17000$$
$$\sum_{i=1}^{18} x_i^2 = 18000\ ,\ \sum_{i=1}^{19} x_i^2 = 19000\ ,\ \sum_{i=1}^{29} x_i^2 = 29000$$
$$\sum_{i=1}^{39} x_i^2 = 39000\ ,\ \sum_{i=1}^{50} x_i^2 = 50000\ ,\ \sum_{i=1}^{60} x_i^2 = 60000$$
$$\sum_{i=1}^{70} x_i^2 = 70000\ ,\ \sum_{i=1}^{80} x_i^2 = 80000\ ,\ \sum_{i=1}^{100} x_i^2 = 100000$$
$$\sum_{i=1}^{120} x_i^2 = 120000\ ,\ \sum_{i=1}^{130} x_i^2 = 130000\ ,\ \sum_{i=1}^{150} x_i^2 = 150000$$
$$\sum_{i=1}^{170} x_i^2 = 170000\ ,\ \sum_{i=1}^{200} x_i^2 = 200000\ ,\ \sum_{i=1}^{220} x_i^2 = 220000$$
$$\sum_{i=1}^{250} x_i^2 = 250000\ ,\ \sum_{i=1}^{280} x_i^2 = 280000\ ,\ \sum_{i=1}^{300} x_i^2 = 312350$$
$$\sum_{i=1}^{330} x_i^2 = 330000\ ,\ \sum_{i=1}^{350} x_i^2 = 350000\ ,\ \sum_{i=1}^{370} x_i^2 = 4579137$$
$$\sum_{i=1}^{400} x_i^2 = 400000\ ,\ \sum_{i=1}^{430} x_i^2 = 430000\ ,\ \sum_{i=1}^{450} x_i^2 = 450000$$
$$\sum_{i=1}^{470} x_i^2 = 470000\ ,\ \sum_{i=1}^{490} x_i^2 = 490000\ ,\ \sum_{i=1}^{500} x_i^2 = 500000$$

The search domain was the Cartesian product of n copies of $\{1, ..., 200\}$, for each n.

Some solutions are shown in Table 1 and, in Table 2, we present the mean number of cost function evaluations spent in the solution for the majority of equations. Finally, Figure 1 shows the graph corresponding to Table 2.

4.2 Equations of the type $x_1^n + x_2^n = N$

The following equations were included in the experiments [2]

$$x_1^2 + x_2^2 = 625$$
$$x_1^3 + x_2^3 = 1008$$
$$x_1^4 + x_2^4 = 1921$$
$$x_1^5 + x_2^5 = 19932$$
$$x_1^6 + x_2^6 = 47385$$
$$x_1^7 + x_2^7 = 4799353$$
$$x_1^8 + x_2^8 = 16777472$$
$$x_1^9 + x_2^9 = 1000019683$$
$$x_1^{10} + x_2^{10} = 1356217073$$
$$x_1^{11} + x_2^{11} = 411625181$$

$$x_1^{12} + x_2^{12} = 244144721$$
$$x_1^{13} + x_2^{13} = 1222297448$$
$$x_1^{14} + x_2^{14} = 268451840$$
$$x_1^{15} + x_2^{15} = 1088090731$$

and the number of cost function evaluations to get to a solution as a function of the equation order is shown in Table 3. The corresponding graph is shown in Figure 2.

5 Conclusion

This paper proposed the application of Fuzzy Adaptive Simulated Annealing to solve Diophantine equations, mainly in cases for which there are no known solutions. According to the presented tests the method was shown to be effective and efficient, being able to solve equations in high dimensional configuration spaces. On the other hand, our figures led to conclusions not exactly compatible with reference [2], especially with respect to the statement that whenever the dimension of state space gets higher, the number of iterations needed to find solutions increases tremendously. Of course, there is an expected increase in cost function evaluations due to combinatorial reasons, but its rate is (surprisingly) not necessarily exponential.

In summary, it is important to highlight that a very important aspect of this type of work is its ability to obtain solutions for a broad spectrum of equations even when we have few theoretical information about them, and making it possible to get valuable information by means of tools apparently not connected to the main subject.

Table 1: Solutions for some examples

Dim.	Solution
12	(55, 9, 11, 5, 3, 2, 2, 3, 1, 5, 23, 3) (20, 4, 6, 17, 2, 14, 1, 1, 13, 41, 5, 32) (1, 2, 1, 50, 11, 1, 4, 1, 33, 2, 10, 2) (1, 5, 5, 2, 2, 3, 4, 4, 30, 15, 4, 51) (61, 3, 4, 1, 1, 2, 4, 7, 2, 1, 4, 2)
13	(56, 2, 1, 67, 3, 11, 54, 13, 45, 2, 6, 3, 9) (17, 6, 9, 37, 5, 16, 36, 25, 3, 65, 54, 33, 28)
14	(56, 28, 20, 17, 2, 44, 53, 39, 11, 49, 19, 6, 11, 9)
15	(18, 18, 2, 31, 29, 37, 8, 1, 92, 13, 17, 24, 1, 13, 38)
16	(9, 1, 63, 74, 3, 13, 20, 1, 1, 7, 15, 17, 64, 8, 27, 21)
17	(38, 7, 87, 2, 2, 1, 3, 24, 7, 7, 48, 2, 5, 68, 12, 12, 1)
18	(49, 79, 4, 51, 43, 3, 6, 5, 1, 15, 5, 3, 27, 2, 10, 58, 19, 2)
19	(45, 10, 10, 37, 58, 6, 6, 11, 36, 3, 57, 6, 26, 28, 13, 1, 11, 72, 18)
29	(21, 3, 6, 25, 6, 29, 30, 29, 20, 71, 29, 2, 55, 34, 3, 10, 33, 25, 2, 17, 1, 92, 36, 13, 30, 28, 1, 32, 7)
39	(11, 14, 14, 50, 10, 14, 3, 11, 39, 18, 32, 30, 31, 79, 6, 50, 17, 14, 25, 27, 15, 23, 6, 46, 2, 44, 22, 25, 8, 30, 5, 14, 52, 4, 21, 23, 6, 95, 18)

References:

[1] S. Abraham and M. Sanglikar, A Diophantine Equation Solver - A Genetic Algorithm Application, *Mathematical Colloquium Journal* 15, 2001.

[2] S. Abraham, S. Sanyal and M. Sanglikar, Particle swarm optimization based Diophantine equation solver, *International Journal of Bio-Inspired Computation* 2, 2010, pp. 100–114.

[3] H. Cohen, *Number Theory - Vol. I.,* Springer–Verlag, New York 2007.

Table 2: Domain dimension x mean number of cost function evaluations spent until convergence (CFE means number of cost function evaluations)

Dim.	CFE	Dim.	CFE	Dim.	CFE
12	5899	39	3211	330	15856
13	5894	50	12732	350	15691
14	3467	60	11943	370	15518
15	3155	70	9432	400	18046
16	5899	80	7813	430	19871
17	2879	100	11875	450	20268
18	3025	120	15371	470	20020
19	2988	130	9753	490	22116
29	2877	150	16443	500	20829

Figure 1: Relation between domain dimension and mean number of cost function evaluations to solve the respective equation

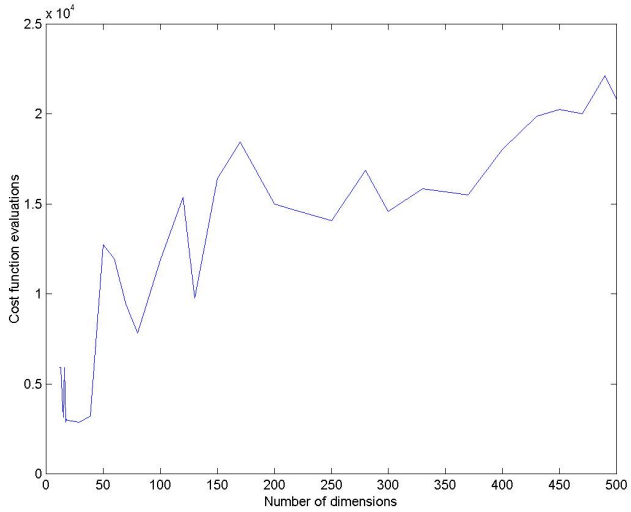
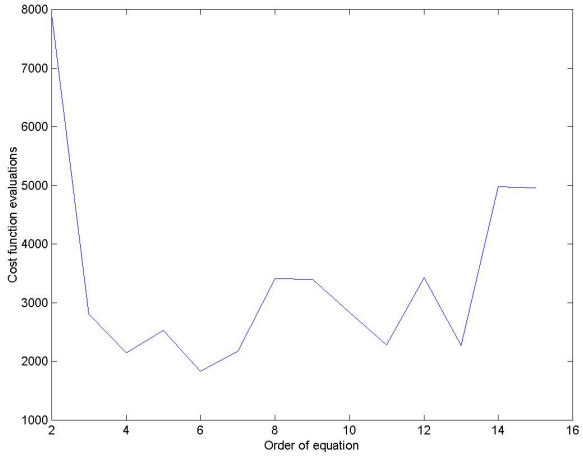


Table 3: Equation order x mean number of cost function evaluations - $x_1^n + x_2^n = N$

Order	# of cost f. e.	Order	# of cost f. e.
2	7900	9	3401
3	2811	10	2837
4	2151	11	2283
5	2533	12	3431
6	1835	13	2275
7	2177	14	4981
8	3412	15	4956

Figure 2: Relation between equation order and average number of cost function evaluations to solve the respective equation - $x_1^n + x_2^n = N$



[4] H. Cohen, *Number Theory - Vol. II.,* Springer–Verlag, New York 2007.
[5] L. Ingber, Adaptive simulated annealing (ASA): Lessons learned, *Control and Cybernetics,* 25 1, 1996, pp. 33–54.
[6] H. Oliveira Jr., Fuzzy control of stochastic global optimization algorithms and VFSR, *Naval Research Magazine* 16, 2003, pp. 103–113.
[7] H. Oliveira Jr. and A. Petraglia, Global optimization using dimensional jumping and fuzzy adaptive simulated annealing, *Applied Soft Computing,* 11, 2011, pp. 4175–4182.
[8] H. Oliveira, Jr., L. Ingber, A. Petraglia, M.R. Petraglia, M.A.S. Machado, *Stochastic Global Optimization and Its Applications with Fuzzy Adaptive Simulated Annealing,* Springer–Verlag, Berlin–Heidelberg 2012.
[9] R. Pachter and Z. Wang, Adaptive Simulated Annealing and its Application to Protein Folding, In *Encyclopedia of Optimization 2nd ed.,* Springer–Verlag, 2009.
[10] B. Rosen, Function optimization based on advanced simulated annealing, In: IEEE Workshop on Physics and Computation - PhysComp '92, 1992, pp. 289-293.
[11] N.P. Smart, *The Algorithmic Resolution of Diophantine Equations,* Cambridge University Press, Cambridge 1998.
[12] H. Szu and R. Hartley, Fast simulated annealing, *Physics Letters A,* 122 (3–4), 1987, pp. 157–162.
[13] P. van Laarhoven, Jr. and E. Aarts, *Simulated Annealing: Theory and Applications,* D. Reidel, Dordrecht, The Netherlands 1987.