Neural, Parallel, and Scientific Computations 18 (2010) 487-498

EXHAUSTIVE VERSUS RANDOMIZED SEARCHES FOR NONLINEAR OPTIMIZATION IN 21ST CENTURY COMPUTING: SOLAR APPLICATION

SYAMAL K. SEN[1] AND GHOLAM ALI SHAYKHIAN[2]

[1]*Department of Mathematical Sciences, Florida Institute of Technology, 150 West University Boulevard, Melbourne, FL 32901-6975, United States*
sksen@fit.edu

[2]*National Aeronautics and Space Administration (NASA), Technical Integration Office (IT-G), Information Technology (IT) Directorate, Kennedy Space Center, FL 32899, United States*
ali.shaykhian@nasa.gov

ABSTRACT We present a simple multi-dimensional exhaustive search method to obtain, in a reasonable time, the optimal solution of a nonlinear programming problem. It is more relevant in the present day non-mainframe computing scenario where an estimated 95% computing resources remains unutilized and computing speed touches petaflops. While the processor speed is doubling every 18 months, the band width is doubling every 12 months, and the hard disk space is doubling every 9 months. A randomized search algorithm or, equivalently, an evolutionary search method is often used instead of an exhaustive search algorithm. The reason is that a randomized approach is usually polynomial-time, i.e., fast while an exhaustive search method is exponential-time i.e., slow. We discuss the increasing importance of exhaustive search in optimization with the steady increase of computing power for solving many real-world problems of reasonable size. We also discuss the computational error and complexity of the search algorithm focusing on the fact that no measuring device can usually measure a quantity with an accuracy greater than 0.005%. We stress the fact that the quality of solution of the exhaustive search – a deterministic method – is better than that of randomized search. In 21st century computing environment, exhaustive search cannot be left aside as an untouchable and it is not always exponential. We also describe a possible application of these algorithms in improving the efficiency of solar cells – a real hot topic – in the current energy crisis. These algorithms could be excellent tools in the hands of experimentalists and could save not only large amount of time needed for experiments but also could validate the theory against experimental results fast.

1. INTRODUCTION

The computing scene has been changing rapidly since 1940's. We have come a long way since then. The processing speed, executable memory storage, hard disk space, as well as band width are practically doubling every year. During 1946-53, the computing speed was about 10^3 operations per sec, during 1953-59 and 1959-64, it was 5×10^4 and

Received April 25, 2010

5×10^5 operations per sec, respectively. During 1964-69 and 1969-75, the computing speed improved to 10^6, 20×10^6 operations per sec, respectively along with the incessant innovations in silicon technology and architecture. During 1975-2010, the speed went on doubling. Today it has reached over 100 teraflops ($100\times10^{12}=10^{14}$ floating-point operations per sec) and touched the petaflops (10^{15} flops) [1]. During most part of twentieth century, it was the main-frame (single) computer used by a large number of users in queues having varying computing resources requirements, which was the dominant factor. As this century was approaching to a close, the main-frame started heading toward oblivion. Its place was being occupied by laptop/desktop computers. Each individual started having his own computer. The dependence on the main frame slowly vanished. Miniaturization, portability, and internet revolutionized the computing scene to such an extent that the whole world will come to a halt if the computer halts. Aircrafts will be grounded, banks will be non-functional, the government departments will cease to function. Besides, we are now having an estimated over 95% computing power unutilized and hence a waste unlike that of the main-frame days.

This enormous computing power available to us has also affected the importance and significance of an algorithm which was earlier considered too slow (exponential-time) and so untouchable. The computer scientists by virtue of their computational complexity study had suggested the user against using combinatorial/ exponential search methods unless the problem is too small. In essence, they have advised against employing these algorithms for real-world problems which are often not too small and also not often too large. In this context, we look back and attempt to take a stock of the situation. We try to use multi-dimensional exhaustive search and find out to which extent it is capable of solving many physical problems, mainly optimization problems. We also compare these search against randomized search to bring out the pros and cons of these two procedures.

In section 2, we state the two types of general nonlinear optimization problems with simple examples. We describe the exhaustive search method as well as a simple randomized search algorithm for nonlinear optimization in this section along with their computational complexities. We consider numerical examples involving simple nonlinear optimization problems of the second type (a more general one) in section 3 while we discuss possible applications for maximizing the efficiency of a solar cell in section 4. Section 5 comprises conclusions.

2. THE PROBLEMS AND THE METHODS

2.1 The problems The nonlinear optimization problems that we will discuss are stated in two different types.

Type 1 Problem Compute $x_i \quad i=1(1)n$ to maximize $f(x_1,x_2,\cdots,x_n)$ subject to $\alpha_i \le x_i \le \beta_i$, where $\alpha_i,\beta_i \quad i=1(1)n$ are numerically specified and the function $f(x_1,x_2,\cdots,x_n)$ are explicitly stated in a non-tabular form.

Type 2 Problem Compute $x_i \quad i=1(1)n$ to maximize $f(x_1,x_2,\cdots,x_n)$ subject to $g_j(x_1,x_2,\cdots,x_n)\le\gamma_j \quad j=1(1)m$, $\alpha_i \le x_i \le \beta_i \quad i=1(1)n$, where $\alpha_i,\beta_i \quad i=1(1)n$ and $\gamma_j, \quad j=1(1)m$ are numerically specified, the functions $g_j(x_1,x_2,\cdots,x_n) \quad j=1(1)m$ and $f(x_1,x_2,\cdots,x_n)$ are explicitly provided in a non-tabular form.

Type 2 problem is more general than Type 1 problem. So we will provide the search methods for Type 2 problem.

2.2 The exhaustive search method The exhaustive search method for both the types of problems divides/dissects the $n-$dimensional space defined by $\alpha_i \le x_i \le \beta_i \quad i = 1(1)n$, i.e., $n-$dimensional rectangular cuboid ($n-$dimensional rectangular parallelepiped or equivalently, $n-$orthotope) – or simply called cuboid here – into an array of smaller cuboids. Let $h_i, i = 1(1)n$ be the length of the $i-$th side of the smaller cuboid. Then $h_i = (\beta_i - \alpha_i)/k_i \quad i = 1(1)n$, where k_i is the number of equispaced divisions of the $i-$th side of the original cuboid. Thus we will have $\prod_{i=1}^{n} k_i$ points, viz.,

$$(\alpha_1, \alpha_2, \cdots, \alpha_n), (\alpha_1 + h_1, \alpha_2, \cdots, \alpha_n), \ (\alpha_1 + 2h_1, \alpha_2, \cdots, \alpha_n), \ \cdots, \ (\alpha_1 + k_1 h_1, \alpha_2, \cdots, \alpha_n),$$
$$(\alpha_1, \alpha_2 + h_2, \cdots, \alpha_n), (\alpha_1, \alpha_2 + 2h_2, \cdots, \alpha_n), \ \cdots, (\alpha_1, \alpha_2 + k_2 h_2, \cdots, \alpha_n),$$
$$\vdots$$
$$(\alpha_1, \alpha_2, \cdots, \alpha_n + h_n), (\alpha_1, \alpha_2, \cdots, \alpha_n + 2h_n), \ \cdots, (\alpha_1, \alpha_2, \cdots, \alpha_n + k_n h_n),$$

at which we compute the value of the function f. We reject those points and their corresponding values of the function f at which one or more constraints $g_j(x_1, x_2, \cdots, x_n) \le \gamma_j \quad j = 1(1)m$ have not been satisfied. We then choose that point out of the remaining points for which the value of f has become maximum. This point is the required approximate solution vector. Call it $x^{(1)}$ – the solution at the first iteration. In order to improve the solution and compute the relative error bounds we repeat the process by halving the length h_i, i.e., by doubling the number equispaced divisions k_i. Consequently we get the improved vector $x^{(2)}$ – the solution at the second iteration. We then obtain the relative error $\| x^{(2)} - x^{(1)} \| / \| x^{(2)} \|$ in the solution vector $x^{(1)}$. Thus, we continue the process till we obtain the desired (relative) accuracy, say 0.5×10^{-4}, in the solution vector $x^{(k)}$, i.e.

$$\| x^{(k+1)} - x^{(k)} \| / \| x^{(k+1)} \| \le 0.5 \times 10^{-4}.$$

2.3 The randomized search method The randomized search algorithm for both the types of problems considers the original cuboid D, generates uniformly distributed p random points ($n-$dimensional vectors) in the original domain $D = \{x_i \mid \alpha_i \le x_i \le \beta_i \quad i = 1(1)n\}$. It computes the function f at these p points. As in the exhaustive search method, we reject those points (out of these p points) and their corresponding values of the function f at which one or more constraints $g_j(x_1, x_2, \cdots, x_n) \le \gamma_j \quad j = 1(1)m$ have not been satisfied. We then choose that point out of the remaining points for which the value of f has become maximum. This point is the required approximate solution vector. Call it $x^{(1)}$ – the solution at the first iteration. To improve the solution and compute the relative error bounds we repeat the process by dividing the domain D into two domains D_1 and D_2, where $D_1 = \{x_i \mid \alpha_i \le x_i \le \beta_i / 2 \quad i = 1(1)n\}$ and $D_2 = \{x_i \mid \beta_i / 2 < x_i \le \beta_i \quad i = 1(1)n\}$. In each

of the two domains, the algorithm generates uniformly distributed p random points in the same way as is done for the original domain. We reject as before those points (out of these $2p$ points) and their corresponding values of the function f at which one or more constraints $g_j(x_1, x_2, \cdots, x_n) \le \gamma_j \quad j = 1(1)m$ have not been satisfied. We then choose that point $(x^{(2)})$ out of the remaining points for which the value of f has become maximum. Consequently we get the improved vector $x^{(2)}$ – the solution at the second iteration. We then obtain the relative error $\| x^{(2)} - x^{(1)} \| / \| x^{(2)} \|$ in the solution vector $x^{(1)}$. Each one of the domains D_1 and D_2 are again subdivided into two domains resulting into four domains $D_{11}, D_{12}; D_{21}, D_{22}$. For each of the four domains, the algorithm generates uniformly distributed p random points in each of these four domains in the same way as is done for the original domain D. Thus we obtain $4p$ points. We reject as before those points (out of these $4p$ points) and their corresponding values of the function f at which one or more constraints $g_j(x_1, x_2, \cdots, x_n) \le \gamma_j \quad j = 1(1)m$ have not been satisfied. We then choose that point $(x^{(3)})$ out of the remaining points for which the value of f has become maximum. Consequently we get the improved vector $x^{(3)}$ – the solution at the third iteration. We then obtain the relative error $\| x^{(3)} - x^{(2)} \| / \| x^{(3)} \|$ in the solution vector $x^{(2)}$. Thus we continue the successive iterations in the same way as above till we obtain the desired accuracy, say 0.5×10^{-4}, in the solution vector $x^{(k)}$, i.e.

$$\| x^{(k+1)} - x^{(k)} \| / \| x^{(k+1)} \| \le 0.5 \times 10^{-4}.$$

The initial number of uniformly distributed random numbers p depends on the specified problem and its dimension. The number p should be reasonably large so that the number of iterations $k+1$ is reasonably small compared to p.

3. NUMERICAL EXAMPLE

We illustrate both the methods using a simple two variable nonlinear optimization problem

$$Max \quad f(x_1, x_2) = -2x_1^{\,2} - x_2^{\,2} + x_1 x_2 + 8x_1 + 3x_2$$

subject to $6x_1 + 2x_2 \le 20$, $2 \le x_1 \le 3, 2 \le x_2 \le 3$.

In *exhaustive search*, we arbitrarily choose 11 points on the x_1-axis, viz., $x_1 = 2(0.1)3$ and 11 points on the x_2-axis, viz., $x_2 = 2(0.1)3$. These constitute 121 pairs of points viz., $(2,2), (2,2.1), \cdots, (3,3)$. Each pair is simply called a point or a vector in the two dimensional space. We evaluate the function $f(x_1, x_2)$ at each of these 121 points. Reject those points that do not satisfy the inequality $6x_1 + 2x_2 \le 20$ and compute the function value for the remaining points and take the largest function value as our approximate solution vector $x^{(1)}$ – the first iteration value. Using the following Matlab commands (in one line)

i=1; for x1=2:.1:3, for x2=2:.1:3, f=-2*x1.^2-x2.^2+x1.*x2+8*x1+3*x2; if (6*x1+2*x2<=20), xx1(i)=x1; xx2(i)=x2; h(i)=f; i=i+1; end; end; end; H=[xx1' xx2' h']; Hsort=sortrows(H, 3), a=size(Hsort); Hsort(a(1),:)

we get x_1, x_2, and the objective function value as $[x_1 \quad x_2 \quad f] = [2.5 \quad 2.5 \quad 15]$. Our $x^{(1)} = [2.5 \quad 2.5]^t$, where t denotes the transpose.

If we now replace .1 in the first line of the Matlab commands by .05 in both the places then we get $[x_1 \quad x_2 \quad f] = [2.45 \quad 2.65 \quad 15.015]$. Our $x^{(2)} = [2.45 \quad 2.65]^t$.

If we now replace .05 in the first line of the Matlab commands by .025 in both the places then we get $[x_1 \quad x_2 \quad f] = [2.475 \quad 2.575 \quad 15.0163]$. Our $x^{(3)} = [2.475 \quad 2.575]^t$.

We continue in this way. When we use the corresponding value .00078125 in the first line of the Matlab commands in both the places then we get $[x_1 \quad x_2 \quad f] = [2.4641 \quad 2.6078 \quad 15.0179]$. Thus our $x^{(8)} = [2.4641 \quad 2.6078]^t$. A true solution is $[x_1 \quad x_2 \quad f] = [2.4643 \quad 2.6071 \quad 15.0179]$

However, one needs to compute after each iteration starting from the second iteration the relative error in the solution vector and proceed until the condition on the relative error in the solution vector $x^{(k)}$, viz., $\| x^{(k+1)} - x^{(k)} \| / \| x^{(k+1)} \| \leq 0.5 \times 10^{-4}$ is satisfied. This satisfaction would imply that the solution vector was correct up to at least 4 significant digits.

We are not presenting here an efficient[1] Matlab code nor are we computing the relative errors after each iteration for the foregoing computations. We are just attempting to show that the algorithm is the simplest and produces the best quality solution. It is, though exponential, has an important role to play in 21st century computing environment for solving numerous practical problems including those connected with laboratory experiments. In this century, the processor speed is doubling every 18 months, band width is doubling every 12 months, and hard disk space is doubling every 9 months. Currently the processing speed is of the order of teraflops (10^{12} floating-point operations per sec) and has touched petaflops (10^{15} floating point operations per sec). In fact, the time for such experiments can be drastically reduced with the aid of such models. There are certain areas that involve truly large optimization problems for which the exhaustive search can be tractable only when the bounds $\alpha_i \leq x_i \leq \beta_i$ are very sharp/narrow. While performing numerical experiments consciously, we do gain significant insight into the problem and consequently could reduce the size of each bound on an element of the solution vector. Such a reduction will enable the exhaustive search find a solution in an acceptable time frame. In fact one has, unlike the main-frame computing days, a complete computer (laptop/desktop) in one's disposal all the time and it is much more powerful compared to those during the 20th century computing devices.

In *randomized search* we generate arbitrarily $p = 100$ uniformly distributed pseudo-random numbers, simply called random numbers, for each element of the solution vector x in each of the intervals $\alpha_i \leq x_i \leq \beta_i \quad i = 1,2$. The Matlab *rand* generates a pseudo-

[1] It can be easily seen that the Matlab code used here computes the function f at half of the points again and again at each successive iteration. This can be avoided while writing an efficient Matlab code. Consequently significant computing time can be saved.

random number r_i in [0,1][2]. To generate a random number x_i in $[\alpha_i, \beta_i]$ from r_i, we compute $x_i = (\beta_i - \alpha_i)r_i + \alpha_i \quad i = 1,2$.Thus using Matlab *rand*, we will generate $100^2 = 10000$ pairs of points or, simply points (x_1, x_2). We evaluate the function $f(x_1, x_2)$ at each of these 10000 points. Reject those points that do not satisfy the inequality $6x_1 + 2x_2 \le 20$ and compute the function value for the remaining points and take the largest function value as our approximate solution vector $x^{(1)}$ – the first iteration value. Using the following Matlab commands (in one line)

>> i=1; for j=1:100, for k=1:100, x1=1+rand; x2=1+rand; f=-2*x1.^2-x2.^2+x1.*x2+8*x1+3*x2; if (6*x1+2*x2<=20), xx1(i)=x1; xx2(i)=x2; h(i)=f; i=i+1; end; end; end; H=[xx1' xx2' h']; Hsort=sortrows(H, 3); a=size(Hsort); Hsort(a(1),:)

we get $x_1, x_2,$ and the objective function value as $[x_1 \quad x_2 \quad f] = [1.9970 \quad 2 \quad 13.9939]$. Our $x^{(1)} = [1.9970 \quad 2]^t$. It can be seen that the *rand* command produces different values every time it is run. This is because of the seed for generation of random numbers, which would be different for every new run. By executing the foregoing commands for the second time we have obtained $[x_1 \quad x_2 \quad f] = [1.9947 \quad 1.9914 \quad 13.9806]$. Our $x^{(1)} = [1.9947 \quad 1.9914]^t$.

However, if we now replace 100 in the first line of the Matlab commands by 200 in both the places then we get $[x_1 \quad x_2 \quad f] = [1.9963 \quad 1.9920 \quad 13.9847]$. Our $x^{(2)} = [1.9963 \quad 1.9920]^t$. We can see that the solution is not improved. Instead it has deteriorated in spite of generating more random numbers. Such a behavior is quite expected when we use *rand* that generates pseudo-random numbers. However, we may replace pseudo-random number generator *rand* by a quasi-random number generator [2-4] since quasi-random numbers are more uniformly distributed with less discrepancy.

If we replace 100 by 1000 (in both the places in the foregoing first line of the Matlab codes) which is significantly much larger, then we should expect an improvement in the solution. The slightly improved solution is $[x_1 \quad x_2 \quad f] = [1.9996 \quad 1.9981 \quad 13.9974]$ which is far from satisfactory. Also, the computing time to evaluate the function f 10 million times besides checking the validity of the inequality $6x_1 + 2x_2 \le 20$ is enormous (over 2 hours) in a 2005 desktop computer.

However, instead of increasing the number of random numbers, one can divide the domain $\alpha_i \le x_i \le \beta_i \quad i = 1,2$ into sub-domains appropriately (not necessarily as mentioned in section 2.3). We evaluate the function $f(x_1, x_2)$ at all the 100 pairs of random points in each of these sub-domains. . Reject those points that do not satisfy the inequality $6x_1 + 2x_2 \le 20$ and retain the remaining points and take that point for which the function value is the largest as our approximate solution in a sub-domain. Having obtained the point corresponding to the largest function value in each domain, we choose the largest of all these values. The vector (point) corresponding to this largest value will

[2] We have not seen Matlab *rand* producing exactly 0 and exactly 1 while generating random numbers. Strictly speaking it generates random numbers in $(0,1)$. However, for our purpose, this does not pose any problem.

be an approximate optimal solution. One can still further subdivide keeping the required random numbers unchanged and continue till a desired accuracy is achieved.

Both the foregoing methods are exponential, the exhaustive search method is deterministic and produce better solution at every successive iteration unlike the randomized search. The genetic algorithms/ evolutionary approaches are always based on the use of a large number of random numbers and are usually polynomial-time. One may certainly go for such an algorithm for obtaining an optimal solution. We are here concerned with scientists/engineers/students who are not well-versed with such algorithms and yet desire to use modeling as an aid to their physical/chemical experiments. The foregoing simple procedures needs no formal knowledge on their part and are easy to program in a high level language such as Matlab/Mathematica.

4. APPLICATION IN MAXIMIZING SOLAR DEVICE EFFICIENCY

Solar devices, particularly solar cells, have not yet become sufficiently competitive and are yet to occupy a significant place among widely used energy sources such as those from fossil fuel (coal, natural gas, diesel, mineral oil). Fossil fuel is the primary source of energy for the world It is increasingly becoming more expensive and straining the economy of most countries of the world while exploration is continuing in obtaining energy from sources such as the sun and the wind and making it competitive. The energy produced from the sun and the wind is increasingly enhanced for our use and it is becoming gradually cheaper. The initial investment is a major hurdle while the later expenditure in producing energy from the sun (or the wind) is significantly low. A situation will crop up in not-so-distant future when the increasing cost of the energy from fossil fuel will be comparable to the decreasing cost of energy from the sun. At that time the solar energy which is available in abundance and is clean (pollution-free) will increasingly start replacing the conventional energy produced by fossil fuel. Currently, many solar energy centers are engaged in maximizing the usage of solar energy available in abundance. Theoretically one can use up to 40% of the energy obtained from the sun while currently we are able to utilize around 15 to 20% of the available solar energy. Hence there is a scope to improve the efficiency of solar devices.

While solar laboratory facilities are increasingly improved and utilized to enhance the effective availability of solar energy, computer modeling can be added along with these facilities to determine the best/optimal values of the concerned parameters fast. Thus such a modeling will cut down the time of experiments with many parameters drastically. In fact, in an experiment, we change only one parameter and observe its effect on another parameter while the rest of the parameters are kept fixed/unchanged. This is because we, the common human beings, are not able to comprehend precisely the effect of simultaneous change of several parameters in an experiment on other parameters. Computer modeling, on the other hand, is capable of doing this job fast and accurately for the experimentalists since the processing speed of the computer has crossed teraflops and has touched petaflops (10^{15} floating-point operations per second) by the end of the first decade of 21^{st} century [1]. This speed is over one thousand folds larger than that during the last decade of 20^{th} century. We present here two forms of global mathematical models for the efficiency of solar cells [5, 6]. Both the models conform to those presented in section 2.1. Hence these can be solved by the exhaustive search method (the simplest possible method).

Model 1: A single multivariate efficiency function model We need to identify the parameters and their optimal achievable values that allow us to increase the efficiency to the maximum possible extent. So far we have identified most of the parameters that contribute to the efficiency. In other words, the efficiency has been found to be the function of the following parameters as described in terms of the notations presented below [5, 6].

Notations

R_p = parallel resistance (∞ for an ideal device)
R_S = series resistance (0 for an ideal device)
J = current density flowing in the device
J_{sc} = short-circuit current
J_0 = reverse saturation current of the diode
$-J_L$ = current generated by light
J_L = reverse current associated with photo-excitation
φ = voltage = forward current device voltage
φ_{oc} = open-circuit voltage
A = area through which current flows
A = ideality factor
$A = 1$ if the transport process is diffusion
$A = 2$ if the transport process is recombination
I = total current = JA (This A is the area unlike the foregoing A which can be identified from the context)
η = efficiency
E_q = semiconductor band gap
ff = fil factor
P_{rad} = total radiation power incident on the cell

The efficiency η is a function of variables/parameters $J_m, \varphi_{oc}, A, k, T, q, \varphi_m$ and possibly J_L, J, J_0, R_s, R_p, *absorption, photon energy*, also. Some of the foregoing parameters appear to be related.

Brief partial sketch of the optimization problem

$$\textit{Maximize} \quad \eta(J_m, \varphi_{oc}, A, k, T, q, \varphi_m, \quad J_L, J, J_o, R_s, R_p, \textit{absorption, photon energy}) =$$
$$(1/P_{rad})[J_m\{\varphi_{oc} - (AkT/q)\ln|(q\varphi_m/AkT)|\}\}$$

subject to

$$\alpha_1 \le J_m \le \beta_1 \quad (\max.\ \textit{possible current density})$$
$$\alpha_2 \le \varphi_{oc} \le \beta_2, \quad \alpha_3 \le A \le \beta_3, \quad \alpha_4 \le k \le \beta_4, \quad \alpha_5 \le T \le \beta_5$$
$$\alpha_6 \le q \le \beta_6, \quad \alpha_7 \le \varphi_m \le \beta_7$$

$\alpha_8 \le J \le \beta_8, \quad \alpha_9 \le J_L \le \beta_9, \quad \alpha_{10} \le J_o \le \beta_{10}$

$\vdots$

$0 \le \varphi_m - \varphi_{oc} + (AkT/q)\ln|(q\varphi_m / AkT) + 1| \le \delta_1 (\textit{small positive value})$

$0 \le \varphi_{oc} - (AkT/q)\ln|(J_L / J_0) + 1| \le \delta_2$ (*another appropriate small value*)

Photon energy versus optical absorption curve provides another inequality

$J - \varphi$ curve provides yet another inequality

$\vdots$

All variables ≥ 0.

Having compiled all the available inequalities, We insert the minimum and maximum attainable values α_i, β_i. These values could be obtained from the laboratory experiments or from literature depicting experimental results. An evolutionary method may be devised to solve the optimization problem. This solution will enable us to determine the best combination of the concerned (attainable) parameter values that will maximize the efficiency of the solar cell. In fact, since there are numerous possible combinations (as it is a combinatorial problem), laboratory experimentation to explore all possible combination is intractable. Through the use of a personal computer which executes over one billion floating-point operations per second (FLOPS), we should be able to get the solution (i.e., the desired optimal parameter values) in a matter of minutes. These values will readily help the laboratory experimentation thus allowing the experimentalists to save significant time on experimentation taking only two parameters at a time while keeping others fixed. Unfortunately the actual situation is not as simple as depicted by the foregoing multivariable efficiency function model. For if it is so, then modeling alone would have saved the enormous trouble of highly time consuming experimentation.

Some of the critical issues for thin film solar cell models are number of layers, band discontinuities, band gaps, and charge in deep bulk states as well as in deep interface states. These need to be appropriately taken care of.

Model 2: Nonlinear programming model involving all possible/available relations We have seen many relations — equations and inequalities — involving thin film pv devices including their concerned materials/compounds. The concerned nonlinear program [6] in a general form can be written as

Maximize efficiency $\eta(x_1, x_2, \cdots, x_n)$ *subject to*

$\mu_i \le or = f_i(x_1, x_2, \cdots, x_n) \le or = \nu_i,\ i = 1(1)m$

(*i.e., all possible equations / inequations so far available to us*)

$\alpha_j \le x_j \le \beta_j,\ j = 1(1)n.$

5. Conclusions

We have described two possible models, viz., Model 1: A single multivariate efficiency function model and Model 2: Nonlinear programming model involving all possible/available relations. The efficiency function in Model 2 is chosen as the function same as in Model 1 or a better one which is (so far) available or developed. Model 2 is definitely better than Model 1 in terms of providing realistic values of the parameters that go to enhance/maximize the efficiency. But computationally it is more involved. If there

is any inconsistency among the relations (equations and inequalities), it would not be possible to get a solution that satisfies all the relations. In such a situation, it is necessary to check and correct the mathematical model for the human mistake committed in modeling. The physical model derived from the material universe is assumed to be errorless (although it need not be).

One important computational aspect is that equality relationships in physical quantities cannot, in general, exist. This is because of the fact that exact physical quantities cannot be exactly represented nor can these be exactly measured. A measuring device is not, in general, more accurate than 0.005% [7]. Consequently we represent almost all equations in terms of inequalities. Even the parameters/variables affecting the efficiency are expressed in the model in terms of their realistic, possibly narrow, bounds as has been done in the first model.

The nonlinear program not being able to/produce a solution implies that the nonlinear program model needs to be corrected/modified based on determining the contradictory inequation(s)/bounds in the model as stated earlier. Such a contradiction (inconsistency) could creep in always due to human mistakes/measuring device errors (accuracy $\leq 0.005\%$).

The later model (Model 2) is certainly not the final model that is the perfect/best one and that can never be improved. Our increasing knowledge about the physics of the solar cell will definitely result in a superior cell in course of time. Thus modeling will continue rather indefinitely in the realm of thin film pv devices. Although the current success of such a modeling is low in terms of aiding the experimentalists as well as reducing the cost and time of experiments, a day will come when we are much better informed through past mistakes/failures. Consequently modeling, we believe, will be the dominant tool in solar cell efficiency arena. However, the later form of the model (Model 2), we believe, is completely general. Only the improvement will occur in the efficiency function and all the two sets of inequalities. While the modeling along with the laboratory experimental activities helps us to gain increasingly deeper insight, the general model will continue to improve in its contents but not in its form [5, 6]. In fact, the models proposed are not static but dynamic in their contents.

The foregoing two models are essentially the ones which can be solved using the methods described in section 2. Specifically, the best method is the exhaustive search described in section 2.2. It can be seen that the way we have presented the method involves computation of the function(s) at the same points again and again. Hence the presented exhaustive search is crude and can be made more efficient by omitting the concerned repetitions. Such omissions need some additional programming effort and is recommended definitely for reasonably large problems. However, for not too large a problem, the exhaustive search described in section 2.2 is not only too easy to be readily coded but also will produce the required optimal solution in an acceptable time frame.

Many models have been proposed in literature [8-28]. All these have varying degree of advantages and scope and are tractable. The exhaustive search for both Models 1 and 2, on the other hand, is the most easily followed and simplest for coding in, say, Matlab by scientists/engineers/students who do not have formal programming knowledge. It is capable of producing the most accurate solution within the precision of computation, but it is exponential-time and intractable if the posed modeling problem is very large. A randomized algorithm based on the usage of quasi-random numbers will not be able to produce as good a solution as the one produced by the exhaustive search. Further we will not be able to deterministically/precisely ascertain, within reasonably narrow bounds, the

quality (relative error bounds) of the solution in a randomized algorithm, probabilistically, this can be determined though.

However, due to the availability of enormous computing power touching petaflops [1] in the current (first) decade of 21st century (see also sections 1 and 3), the exponential-time exhaustive search is not altogether untouchable if the posed real-world modeling problem is not very large. In fact we do have many small as well as large real-world problems for which we have not tried exhaustive search consciously possibly thinking about its combinatorial computational (exponential) complexity. Now it is slowly entering into the realm of computation not only because of ever increasing computing power [7] but also because of (i) availability of individual computer (lap-top/desktop) and (ii) non-utilization of an estimated 95% computing resources (power) unlike main-frame era in mid-/late 20th century.

REFERENCES

1. Chidanand Rajghatta (2007). The Times of India, TNN, India hosts world's fourth fastest supercomputer, The Times of India Daily News Paper, reported on November 13, 2007 at 2143 hours Indian Standard Time from Washington.

2. S.K. Sen, T. Samanta and A. Reese (2006). Quasi- versus pseudo-random generators: discrepancy, complexity and integration-error based comparison, *International Journal of Innovative Computing, Information and Control*, **2**, No. 3, pp. 621-651, June 2006.

3. V. Lakshmikantham, S.K. Sen, and T. Samanta (2005). Comparing random number generators using Monte Carlo integration, *International Journal of Innovative Computing, Information and Control*, **1**, No. 2, pp. 143-165, June 2005.

4. S.K. Sen and G.A. Shaykhian (2007). Scope of various random number generators in ant system approach for TSP Paper AC2007-458, 2007 ASEE (American Society for Engineering Education) Proc. Annual Conference & Exposition, Hilton Howaii Village, Honolulu, Howaii, June 24-27, 2007, 1-25.

5. S.K. Sen(2007). Efficiency of solar cell: scope of modeling in experimental environment (an invited talk), *Fifth International Conference on Dynamic Systems and Applications* (mat 30-June 02, 2007), Atlanta, Georgia (supported by Florida Solar Energy Center award/grant # 211064).

6. S.K. Sen (2009). How modeling can attract experimentalists to improve solar cell's efficiency: Divide-and-conquer approach, *Nonlinear Analysis*, 10.1016/j.na.2008.10.058, October, 2008, **71** (2009), 196-211.

7. V. Lakshmikantham and S.K. Sen (2005): Computational Error and Complexity in Science and Engineering, Elsevier, Amsterdam.

8. Schwarz, R., Gray J., Turner, G. Kanani, D., and Ullal, H. (1984). Numerical modeling of $p-I-n$ hydrogenated thin film silicon cells, *Conference Record, 17th IEEE Photovoltaic Specialists Conference*, Kissimmee, Florida, May, 1984, 369-373.

9. Dimmer, B., Dittrich, H., Menner, R., and Schock, H.W. (1987). Performance and optimization of heterojunctions based on Cu(Ga,In)Se_2, *Conference Record, 19th IEEE Photovoltaic Specialists Conference,* New Orleans, May 1987, 1454-1460.

10. Basore, P.(1990). Numerical modeling of textured silicon solar cells using PC-ID, *IEEE Trans. Elecron Devices*, **37,** 2, 337.

11. Gray, J.L. (1991). ADEPT: A general purpose device simulator for modeling solar cells in one, two and three dimensions, *Conference Record, 22nd IEEE Photovoltaic Specialists Conference*, Las Vegas, NV, 1991, 436-438.

12. Lee, Y.J. and Gray, J.L. (1993). Numerical modeling of polycrystalline CdTe and CIS solar cells, *Conference Record, 23rd IEEE Photovoltaic Specialists Conference*, Luisville, May 1993, 586-591.

13. Lee, Y.J. and Gray J.L. (1994). Numerical modeling of the temperature and illumination intensity dependent performance of CIS solar cells, *Proceedings of the 12th European Photovoltaic Solar Energy Conference*, Amsterdam, April 1994, 1561-1564.
14. Gray, J.L. (1996). Interpretation of capacitance-voltage characteristics in thin film solar cells using a detailed numerical model, *Conference Record, 25th IEEE Photovoltaic Specialists Conference*, Washington D.C., April 1996, 905-908.
15. Niemegeers, A. and Burgelman, M. (1996). Numerical modeling of ac-characteristics of CdTe and CIS solar cells, *Conference Record, 25th IEEE Photovoltaic Specialists Conference*, Washington D.C., April 1996, 901-904.
16. Clugston, D. and basore, P. (1997). PCID version 5: 32-bit solar cell modeling on personal computers, *Conference Record, 26th IEEE Photovoltaic Specialists Conference*, Anaheim,
17. Burgelman, M., Nollet, P., Degrave, S. and Beier, J. (2000). Modeling the crossover of the I-V characteristics of thin film CdTe solar cells, *Conference Record, 28th IEEE Photovoltaic Specialists Conference*, Anchorage, September 2000, 551-554.
18. Fahrenbruch, A. (2000). Modeling results for CdS/CdTe solar cells, *Technical Report*, March 2000, Colorado State University.
19. Burgelman, M., Nollet, P., Degrave, S.(2000). Modeling polycrystalline semiconductor solar cells, *Thin Solid Films*, 361-362, 527-532.
20. Grasso, C., Ernst, K., Konenkamp, R. Burgelman, M., Lux-Steiner, M-C. (2001). Photoelectrical characterization and modeling of the eta-solar cell, *Proceedings of the 17th European Photovoltaic Conference*, Munich, October 2001, 211-214.
21. Klenk, R. (2001). Characterization and modeling of chalcopyrite solar cells, *Thin Solid Films*, 387, 135-140.
22. Dullweber, T., Hanna, G., Rau, U. and Schock, H.W. (2001). A new approach to high-efficiency solar cells by band gap grading in Cu(In, Ga)Se_2 chalcopyrite semiconductors, *Solar Energy Materials and Solar Cells*, **67**, 145-150.
23. Fahrenbruch, A. (2002). Comparison of experimental data with AMPS modeling of the effects of CdS layer thickness on the CdS/CdTe solar cell, *Conference Record, 29th IEEE Photovoltaic Specialists Conference*, New Orleans, May 2002, 551-554.
24. Huang, C.H., Li, S.S. and Anderson, T.J. (2002). Device modeling and simulation of CIS-based solar cells, *Conference Record, 29th IEEE Photovoltaic Specialists Conference*, New Orleans, May 2002, 748-751.
25. Burgelman, M. and Grasso, C.(2003). Flatband solar cells: a model for solid-state nano-structured solar cells, Presented at the *3rd World Conference of Photovoltaic Energy Conversion*, Osaka, May 2003.
26. Gloeckler, M., Fahrenbruch, A.L. and Sites, J.R. (2003). Numerical modeling of CIGS and CdTe solar cells: Setting the baseline, *3rd World Conference of Photovoltaic Energy Conversion*, Osaka, May 2003.
27. Bube, R. H. (1998). *Photovoltaic Materials*, Imperial College Press, London.
28. Burgelman, M. Verschraegen, J., Degrave, S., and Nollet, P. (2004). Modeling thin-film pv devices, *Prog. Photovolt: Res. Appl.* **12**, 143-153.