An improvement on Fibonacci search method in optimization theory

Article *in* Applied Mathematics and Computation · January 2004

DOI: 10.1016/S0096-3003(02)00828-7

CITATIONS
29

READS
4,911

3 authors, including:

Necmettin Yildirim
New College of Florida
30 PUBLICATIONS **775** CITATIONS

SEE PROFILE

Bünyamin Yildiz
Mustafa Kemal University
27 PUBLICATIONS **186** CITATIONS

SEE PROFILE

Available at
www.ElsevierMathematics.com
POWERED BY SCIENCE DIRECT®

ELSEVIER Applied Mathematics and Computation 147 (2004) 893–901

APPLIED MATHEMATICS AND COMPUTATION

www.elsevier.com/locate/amc

An improvement on Fibonacci search method in optimization theory

Murat Subasi [a,*], Necmettin Yildirim [b], Bünyamin Yildiz [a]

[a] *Fen Edebiyat Fakültesi, Matematik Bölümü, Atatürk Üniversitesi, 25240 Erzurum, Turkey*

[b] *Atatürk Üniversitesi, Bilgisayar Bil. Uyg. ve Arastirma Merkezi, 25240 Erzurum, Turkey*

Abstract

Among the other elimination methods, Fibonacci search method is regarded as the best one to find the optimal point for single valued functions. In the present study, employing Lucas numbers instead of Fibonacci numbers, we have made partial improvements on location of the intervals that contain optimal point in the Fibonacci search algorithm. For that purpose, using two well known test functions in optimization theory, a computer program was developed in MAPLE to examine this idea. It was seen that the improved method is giving better results than the previous method in the sense that converging the optimal point more rapidly.

Keywords: One-dimensional optimization; Unimodal function; Fibonacci numbers; Lucas numbers

1. Introduction

The classical methods of optimization are useful in finding the optimum of continuous and differentiable functions. These methods are analytical and make use of the techniques of differential calculus in locating the optimum points. Since some of the practical problems raising in many scientific areas involve objective functions that are not continuous and differentiable, the

* Corresponding author.
E-mail address: msubasi@atauni.edu.tr (M. Subasi).

0096-3003/$ - see front matter © 2002 Elsevier Inc. All rights reserved.
doi:10.1016/S0096-3003(02)00828-7

classical optimization techniques have limited scope in practical applications. However, a study of the calculus methods of optimization forms a basis for developing most of the numerical techniques of optimization. The differential calculus method of optimization is an analytical approach and is applicable to continuous and twice differentiable functions. In this method, the calculation of the numerical value of the objective function is virtually the last step of the process. The optimal value of the objective function is calculated after determining the optimal values of the decision variables.

In the numerical methods of optimization as in Fibonacci search method, an opposite procedure is followed in that the values of the objective function are first found at various combinations of the decision variables and conclusions then drawn regarding the optimal solution [1,2].

2. The Fibonacci search method

A unimodal function is one that has only one peak(maximum or minimum) in a given interval. Thus, a function of one variable is said to be unimodal if, given that two values of the variable are on the same side of the optimum, then the one nearer the optimum gives the better functional value, i.e., the smaller value in the case on a minimization problem.

Thus a unimodal function can be a nondifferentiable or even a discontinuous function. If a function is known to be unimodal in a given range, the interval in which the minimum lies can be narrowed down provided the function values are known at two different points in the range. The assumptions of unimodality are made in all the elimination techniques. If a function known to be multimodal the range of the function has to be subdivided into several parts and each part treated separately as a unimodal function.

In order to apply the Fibonacci search method in a practical problem, the following criteria's must be satisfied:

(i) The initial interval of uncertainty, in which the optimum lies, has to be known.
(ii) The function being optimized has to be unimodal in the initial interval of uncertainty.
(iii) The exact optimum cannot be located in this method. Only an interval, known as the final interval of uncertainty will be known. The final interval of uncertainty can be made as small as desired by making more computations.
(iv) The number of function evaluations to be used in the search or the resolution required has to be specified beforehand.

As it is known, the Fibonacci numbers are defined by $F_0 = 0$, $F_1 = 1$, $F_{n+2} = F_{n+1} + F_n$, i.e.,

n	0	1	2	3	4	5	6	7	8	...
F_n	0	1	1	2	3	5	8	13	21	...

and the Lucas numbers are defined by $L_0 = 2$, $L_1 = 1$, $L_{n+2} = L_{n+1} + L_n$, i.e.,

n	0	1	2	3	4	5	6	7	8	...
L_n	2	1	3	4	7	11	18	29	47	...

3. Improved Fibonacci search algorithm

In this section, we will investigate the solution of the

$$\min_{x\in[a,b]} f(x) \to \min$$

minimization problem, where the function $f(x)$ is unimodal on the interval. Let the number of iteration, the interval that involving optimal point and the objective function be given. Let L_k denote kth Lucas number. Then our algorithm proceeds as follows:

Step 1. Initialize a, b, $f(x)$.
Step 2. From $k = 1$ to n do.
Step 3. Choose as $[a_1, b_1] = [a, b]$ and take $k = 1$.
Step 4. Calculate the points

$$x_k = a_k + (b_k - a_k)\frac{L_{n-k+1}}{L_{n-k+3}}, \qquad x'_k = a_1 + (b_1 - a_1)\frac{L_{n-k+2}}{L_{n-k+3}}$$

Step 5. Calculate the values $f(x_k)$ and $f(x'_k)$.
Step 6. If $(x_k) \leqslant f(x'_k)$ then $[a_{k+1}, b_{k+1}] = [a_{k+1}, x'_k]$. Otherwise (if $f(x_k) > f(x'_k)$) $[a_{k+1}, b_{k+1}] = [x_k, b_{k+1}]$.
Step 7. If $k + 1 \neq n$ then go to **Step 4**, otherwise stop the iteration.

Throughout the steps above, the points x_k and x'_k will be symmetrical according to the middle point of the interval $[a_k, b_k]$ and the minimum point x^* will be the element of the interval $[a_k, b_k]$ for $k = 1, \ldots, n$. In each iteration (for $2 \leqslant k \leqslant n$), length of the interval is given by

$$b_k - a_k = (b - a)\frac{L_{n-k+3}}{L_{n+2}}$$

and the last interval is located as

$$x_n = a_n + (b_n - a_n)\frac{L_1}{L_3}, \qquad x'_n = a_n + (b_n - a_n)\frac{L_2}{L_3}$$

and the value of the objective function at optimal point $\hat{x}$ is calculated by

$$f(\hat{x}) = f\left(a_n + \frac{(b_n - a_n)}{2}\right)$$

The error of the improved method is given by the following estimate:

$$|\hat{x} - x^*| \leqslant \frac{1}{2}(b_n - a_n) = t(b-a)\frac{1}{L_{n+2}}$$

4. Test problems to compare the improved method with classical Fibonacci search method

In order to compare the classical Fibonacci search method with the improved one on two well known test problems, A MAPLE procedure of which the source code was given in Appendix A, was developed [3]. The code was developed in MAPLE V Release 4.0a.

Example 1. Let us consider the function [4]

$$f(x) = \left|\frac{(x-1)}{4}\right| + \left|\sin\left(\pi\left(1 + \frac{(x-1)}{4}\right)\right)\right| + 1$$

on the interval $[-3, 3]$. Minimum point for this function is $x^* = 1.0$. Minimum value of this function is $f(x^*) = 1.0000$ (Fig. 1).

Results of both the improved and classical Fibonacci search method for a number of iteration were given in Table 1.

Example 2. Now, we will consider the function [5]

$$f(x) = -\sum_{i=1}^{10} \frac{1}{(k_i(x - a_i))^2 + c_i}$$

where

$$a = [4.496, 4.885, 0.800, 4.986, 3.901, 2.395, 0.945, 8.371, 6.181, 5.713]$$
$$k = [2.871, 2.328, 1.111, 1.263, 2.399, 2.629, 2.853, 2.344, 2.592, 2.929]$$
$$c = [0.149, 0.166, 0.175, 0.183, 0.128, 0.117, 0.115, 0.148, 0.188, 0.198]$$

on the interval $[4, 5.2]$. Minimum point for this function is $x^* = 4.855$. Minimum value of this function is $f(x^*) = -13.92234$ (Fig. 2).

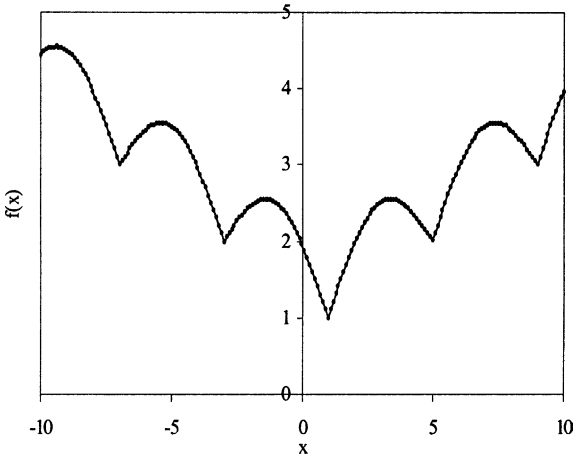

Fig. 1. Graphical demonstration of the objective function given in Example 1.

Table 1
Results of the improved method for the test objective function given in Example 1 for various number of iteration of $n = 2$, 4, 10, 20, and 30

Number of iteration, n	Last interval determined		Minimal value of objective the function	
	Classical method	Improved method	Classical method	Improved method
2	[−1.00000, 3.00000]	[−0.42857, 3.00000]	2.50000	1.29394
4	[0.00000, 1.50000]	[0.33333, 1.66666]	1.50768	1.00000
10	[0.95833, 1.04166]	[0.96894, 1.04347]	1.04313	1.00643
20	[0.99954, 1.00046]	[0.99969, 1.00030]	1.00023	1.00000
30	[0.99999, 1.00000]	[0.99999, 1.00000]	1.00000	1.00000

Results of both the improved and classical Fibonacci search method for a number of iteration were given in Table 2.

5. Results and discussion

In one-dimensional optimization, one of the best algorithms is Fibonacci method's algorithm. In this study, we have made improvements on Fibonacci search method, which gives better results than classical Fibonacci search method. Furthermore, when locating involved optimal point by Lucas numbers instead of Fibonacci numbers in Fibonacci search algorithm, the

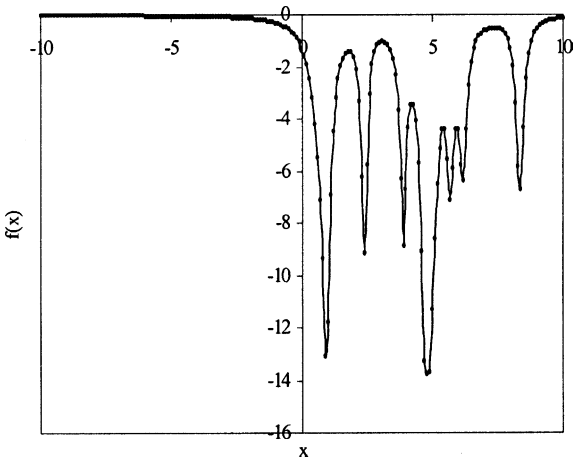

Fig. 2. Graphical demonstration of the objective function given in Example 2.

Table 2
Results of the improved method for the test objective function given in Example 2 for various number of iteration of $n = 2, 5, 8, 10$, and 15

Number of iteration, n	Last interval determined		Minimal value of objective the function	
	Classical method	Improved method	Classical method	Improved method
2	[4.40000, 5.20000]	[4.51428, 5.20000]	−6.51613	−13.92210
5	[4.73846, 4.92307]	[4.74482, 4.91034]	−13.73688	−13.86864
8	[4.82909, 4.87272]	[4.83902, 4.87804]	−13.88974	−13.92174
10	[4.85000, 4.86666]	[4.84968, 4.86459]	−13.91953	−13.92210
15	[4.85510, 4.85660]	[4.85286, 4.85623]	−13.92239	−13.92234

improved method gives better results. It was confirmed by two well known optimization test problems and the results were given in Tables 1 and 2. As it can be seen form the tables, length of the last intervals computed by improved method for each number of iteration are smaller than Fibonacci search method. Moreover, the optimal values of the objective functions are more sensitive in the improved method.

Some of the optimization methods for multivariate functions, such as Powell's method, consist of one-dimensional search algorithms in order to converge to the optimum points. Since the improved method converges to the optimal points more rapidly and outputs more accurate value of the objective function at that point, it is expected that the improved method will make crucial contribution to such methods especially when applied large scaled

optimization problems in terms of computation times and accurateness of the objective function value at optimal points.

Appendix A. MAPLE source code to find optimal points of a given unimodal function using Fibonacci search method and the improved method

```
Improved := proc(f::procedure, Lucas::procedure, n, a0, b0)
# n          : number of iteration
# a0         : begining point of the starting interval
# b0         : end point of the starting interval
# f(x)       : The objective function
# minFibl    : Minimum value of the function at starting point
               of the last interval computed by Fibonacci
               Search Method
# minFib2    : Minimum value of the function at ending point
               of the last interval computed by Fibonacci
               Search Method
# minLucas   : Minimum value of the function computed by The
               Improved Method


local a, b, x, k, minFibl, minFib2, minLucas;

# Loading MAPLE's combinat package to Generate Fibonacci
  numbers
with (combinat, Fibonacci):
a := array(1...n+1) : b := array(1...n+1) :
x := array(1...n+1, 1...2) :
a[1] := a0 : b[1] := b0 :

#- - - - - -Fibonacci Search Method- - - - - -
for k from 1 to n do
  x[k,1] := a[k] + (b[k] - a[k]) * Fibonacci(n-k+1)/
  Fibonacci(n-k+3) :
  x[k,2] := a[k] + (b[k] - a[k]) * Fibonacci(n-k+2)/
  Fibonacci(n-k+3) :
     if evalf(f(x[k,1])) <= evalf(f(x[k,2])) then
       a[k+1] := a[k] :
       b[k+1] := x[k,2] :
     else
       a[k+1] := x[k,1] :
       b[k+1] := b[k] :
```

```
     fi:
od:
minFib1 := evalf(f(b[k - 1])) :
minFib2 := evalf(f(a[k - 1])) :
lprint(evalf(a[k - 1]), evalf(b[k - 1]));

#- - - - - -The Improved Method- - - - - -
for k from 1 to n do
x[k, 1] := a[k] + (b[k] - a[k]) * Lucas(n - k + 1)/Lucas(n - k + 3) :
x[k, 2] := a[k] + (b[k] - a[k]) * Lucas(n - k + 2)/Lucas(n - k + 3) :
if evalf(f(x[k, 1])) <= evalf(f(x[k, 2])) then
    a[k + 1] := a[k] :
    b[k + 1] := x[k, 2] :
else
    a[k + 1] := x[k, 1] :
    b[k + 1] := b[k] :
fi:
od:
lprint(evalf(a[k - 1]), evalf(b[k - 1]));
if minFib1 < minFib2 then
  lprint(minFib1):
else
    lprint(minFib2);
fi:
minLucas := evalf(f(a[k - 1] + (b[k - 1] - a[k - 1])/2.)) :
lprint(minLucas);
end:

#———Lucas Numbers Generator Procedure———
Lucas := proc(n)
local i;
Lucas(0) := 2 :
Lucas(1) := 1 :
for i from 2 to n do
  Lucas(i) := Lucas(i - 1) + Lucas(i - 2) :
od:
end:
```

References

[1] P.E. Gill, W. Murray, M.H. Wright, Practical Optimization, Academic Press, London, 1981.
[2] B.D. Sivazlian, L.E. Stanfel, Optimization Techniques in Operations Research, Prentice-Hall, NJ, 1975.

[3] R. Nicolaides, N. Walkington, MAPLE: a comprehensive introduction, Cambridge University Press, NY, 1996.
[4] D. Ratz, A nonsmooth global optimization technique using slopes: the one-dimensional case, J. Global Optim. 14 (4) (1999) 365–393.
[5] A. Törn, A. Zilinskas, Global Optimization, Lecture Notes in Computer Science, no. 350, Springer-Verlag, Berlin, 1989.