S.T.E.P. : The Easiest Way to Optimize a Function

Stefan LANGERMAN
Grégory SERONT
Hugues BERSINI

Technical Report No.

TR/IRIDIA/94-7

To appear in : Proceedings of the IEEE World Congress on Computational Intelligence, WCCI'94, June 26 - July 22, 1994, Florida, U.S.A.

S.T.E.P. : The Easiest Way to Optimize a Function

Stefan Langerman false Swarzberg[1],
email : slangerm@ulb.ac.be
Grégory Seront[1], email : gseront@ulb.ac.be
Hugues Bersini[2], email : bersini@ulb.ac.be

1. Département d'Informatique, Université Libre de Bruxelles, CP 212, Av. Franklin Roosevelt 50, 1050 Bruxelles
2. IRIDIA, Université Libre de Bruxelles, CP 194/6, Av. Franklin Roosevelt 50, 1050 Bruxelles

Abstract — Most of the algorithms for global optimization making use of the concept of population exploit very little of the information provided by agents in the population in order to choose the next point to evaluate. In this paper, we develop a new method called S.T.E.P. (Select The Easiest Point) which determines the next point to evaluate by analysing the usefulness of evaluating the function at a certain position. Moreover, we will see that this method permits one to precisely define the heuristic of the search. We will also prove its convergence under certain conditions.

I.Introduction

Most of the algorithms for global optimization making use of the concept of population (such as Genetic Algorithms [7], Evolution Strategies [1], Sampling and Clustering, MultiGreedy [6], etc.) exploit very little of the information provided by agents in the population in order to choose the next point to evaluate, and thus, misuse the knowledge gleaned during the previous evaluations of the function to optimize.

The first attempts to solve this problem were made by the Immune Recruitment Mechanism [4]. The primary idea was to determine certain *zones of recruitment*, where it would be a good idea to choose the next evaluations. The zones proposed then were regions *near* the good points, and *far* from the bad ones. This approach seems to have given some promising results, but unfortunately is too vague and too complex to be analysed formally.

In this paper, we develop a new method to determine the next point to evaluate by analysing the usefulness of evaluating the function at a certain position. Moreover, our choice will be based on the value of all the known points of the search space.

The basic idea is that we want to evaluate the function at the poir for which we have the greatest chance of exceeding the *best poi* found until then. To simplify the method, we will divide all th search space into *partitions* delimited by the known points, an determine the *partition* having the greatest chance of including point which exceeds the *best point*.

In order to evaluate the difficulty of exceeding the *best point* in particular *partition* of the space, we will have to define the *diff culty of a function*. This *difficulty* can be any scalar value whic gives some information about the number of function evaluatio required to find the global optimum of the function considered. has in general to be closely related with a notion of neighbou hood.

The first expression of the *difficulty* will be inspired from son calculations by Richard P. Brent [5]. This is a further illustration the benefits gained by hybridizing an evolutionary type of sear for its exploration strength with classical optimization methods f their exploitation strength, a viewpoint largely debated in [3].

We can now define the *difficulty of a partition* as the difficulty the easiest function passing through the points delimiting this pa ticular *partition*, which exceeds the *best point* known. This ea ness will be defined in section II.

Note that this *difficulty* notion is highly suited to identifying t most promising recruitment zones. Also, it defines in a rath direct way the heuristic of search.

In this paper, we will restrict our discussion to global optimizat of one dimensional real function. Extensions to *n* dimensions a combinatorial spaces are under study.

the next section, we define the *difficulty of a function*, and show w to derive the *difficulty of a partition* of the space.

en, we present the structure of the algorithm, and prove its con- ·gence. Finally, we will show some encouraging experimental :ults.

II. Difficulty of a Segment

t $f(x)$ be a function of one parameter $x \in [a,b]$ to maximize, d suppose that we have already evaluated $f(x)$ at the points , $x_1, \ldots, x_n$, with $x_0 = a$, $x_n = b$, and $x_i < x_{i+1}$. Let $f_i = f(x_i)$ the value of $f(x)$ for those points.

e can divide the search space $[a,b]$ into n segments delimited by : x_i :

$$I_i = [x_{i-1}, x_i],\ i = 1,\ldots,n \tag{EQ 1}$$

ie next problem is to define the *difficulty of a function*. Our defi- .ion will be inspired by the fact that [5] :

If we have a bound

$$\|f''(x)\|_\infty \le M \tag{EQ 2}$$

then, the maximum number of function evaluations required to find the optimum value of $f(x)$ within some prescribed tolerance t is of order $\sqrt{M/t}$.

e can easily see that if M is big, we will have to wait longer than it were small. Therefore, this bound on the second derivative can rve us as a definition of the *difficulty of a function*.

› calculate the difficulty D of a segment I_i, we have to search for e function with the lowest M (the easiest) which passes through e two extremities of the segment, and exceeds the value of the *st point* found until then. Brent [5] demonstrates that this func- on is a parabola.

› simplify the calculations, we can translate the coordinates so as place the origin at the left extremity (x_{i-1}, f_{i-1}) of the segment. :t

$$\Delta x = x_i - x_{i-1}$$
$$\Delta y = f_i - f_{i-1}$$
$$\hat{y} = f^* - f_{i-1} + t$$

'here

$$f^* = \max(f_i) \tag{EQ 3}$$

id t is the tolerance (i.e. the precision required for the value of ie optimum). We have to add t because we are only interested in oints which exceed the value of the best point by this tolerance or iore.

So, we will build a parabola passing through the two extremities of the segment $(0,0)$ and $(\Delta x, \Delta y)$, and whose value at the optimum is $\hat{y}$. We will try then to determine the (absolute) value D of the second derivative of this parabola, which is the difficulty of the segment considered.

Consider the parabola passing through the point $(0,0)$.

$$y = -\frac{D}{2}x^2 + bx \tag{EQ 4}$$

with $D \geq 0$. If we want this equation to pass through the point $(\Delta x, \Delta y)$, then

$$b = \frac{\Delta y}{\Delta x} + \frac{D}{2} \cdot \Delta x \tag{EQ 5}$$

To find the optimum of the parabola, we determine the first derivative of y.

$$y' = -Dx + b \tag{EQ 6}$$

The value of x at the optimum of the parabola is

$$\hat{x} = \frac{1}{D} \cdot b \tag{EQ 7}$$

We can inject this value in (EQ 4), knowing that $y = \hat{y}$ for $x = \hat{x}$ (the value we need at the optimum in order to exceed the best point so far by a tolerance of t).

$$\hat{y} = -\frac{D}{2} \cdot \frac{1}{D^2} \cdot b^2 + \frac{1}{D} \cdot b^2 = \frac{1}{2 \cdot D} \cdot b^2 \tag{EQ 8}$$

Combining (EQ 8) and (EQ 5), we have

$$\frac{\Delta x^2}{4} \cdot D^2 + (\Delta y - 2\hat{y}) \cdot D + \frac{\Delta y^2}{\Delta x^2} = 0 \tag{EQ 9}$$

Solving the equation for D, we find :

$$D = \frac{4\hat{y} - 2 \cdot \Delta y \pm 4\sqrt{\hat{y}^2 - \hat{y} \cdot \Delta y}}{\Delta x^2} \tag{EQ 10}$$

We only keep the solution for which $\hat{x}$ is in the segment considered. Then, the final expression of the difficulty of a segment is:

$$D = \frac{4\hat{y} - 2\Delta y + 4\sqrt{\hat{y}^2 - \hat{y} \cdot \Delta y}}{\Delta x^2} \tag{EQ 11}$$

III.The Algorithm

In order to evaluate the performances of methods based on our notion of difficulty, we present here a very simple algorithm implementing the concepts presented above.

The idea is to start with an unique segment delimited by the bounds of the search space (i.e. $[a,b]$). At each iteration, we select the easiest segment (the one with the lowest D), and evaluate the point at its center. This new point divides the segment into two new segments.

```
Evaluate $f(x)$ at the points $a$ and $b$
Initialize the first segment $[a,b]$
while (stop criterion) do
    Determine the segment with
        the lowest $D$
    Evaluate the point at the center of this
        segment
end while
```

IV.Convergence

In this section, we will determinate an upper bound on the number of evaluations required in order to find the global optimum with a tolerance t, knowing an upper bound on the second derivative of the function considered.

Suppose that the second derivative of $f(x)$ exists and is bounded by M:

$$\|f''(x)\|_{\infty} \leq M \tag{EQ 12}$$

In order to find an upper bound on the number of segments necessary to obtain the global optimum with a desired precision t, it might be interesting to look at the size of the biggest segment in which we are sure that no better point can be found. To be certain, we must have:

$$M \leq D \tag{EQ 13}$$

Remember that D (EQ 11) is the minimal value of the second derivative needed in order to exceed the value of the best point found until then by a tolerance t. (EQ 13) transforms into:

$$M \leq \frac{4\hat{y} - 2\Delta y + 4\sqrt{\hat{y}^2 - \hat{y}\cdot\Delta y}}{\Delta x^2} \tag{EQ 14}$$

Without any loss of generality, we can suppose that $f_{i-1} \geq f_i$ (i.e. $\Delta y \leq 0$). Since $M > 0$, we can transform the (EQ 14) in the following way :

$$\Delta x^2 \leq \frac{1}{M} \cdot (4\hat{y} - 2\Delta y + 4\sqrt{\hat{y}^2 - \hat{y}\cdot\Delta y}) \tag{EQ 15}$$

Since $\Delta x > 0$, we see that the bound over Δx is growing with $\hat{y}$ and decreasing with Δy, and at worse, we have $\hat{y} = t$, an $\Delta y = 0$. The minimum segment size is:

$$\Delta x_{min} = \sqrt{\frac{8 \cdot t}{M}} \tag{EQ 1}$$

And the maximum number of segments is the smallest power of greater than

$$\left\lceil \sqrt{\frac{M}{8 \cdot t}} \cdot (b - a) \right\rceil \tag{EQ 1}$$

And the maximum number of function evaluations is this numb plus one.

Note that the value of (EQ 17) +1 corresponds to the upper bour on the minimum number of function evaluations required by t well-known algorithm *Glomin* of Richard P. Brent [5]. *Glomin* presented in the literature [10] as the best algorithm which guara tees giving the global optimum of a function (within some pr scribed tolerance).

A fundamental difference between Glomin and our method is th Glomin requires an upper bound on the second derivative of t function to optimize as an *à priori* condition in order to wo properly. Our algorithm does not require this knowledge, and c provide, at each iteration, a bound on the second derivative f which we are sure of having obtained the global optimum.

In a way, while Glomin needs an *à priori* condition, our meth gives the same condition, but *à posteriori*, after each iteration.

V.Experiments

We have tested the algorithm presented above on a set of test fu tions taken from the literature. Note that in some cases, we ha had to modify them a bit in order for them to be usable in o dimension

In each case, we have compared our algorithm S.T.E.P. with *Ge sis*, a Genetic Algorithm developed by John Grefenstette, a often used as a standard of comparison. We also compared method with *Glomin*, which is one of the best known algorith for global opimization presented in the Numerical Analysis lite ture.

As *Glomin* requires an upper bound on the second derivative the function to optimize in order to work correctly, we will c ment, in each case, the value used for this parameter.

In this article, we consider the performance of *Genesis* to be average of 10 independent runs (on a logarithmic scale).

We will present for each function, a graph of performance sh ing the error in logarithmic scale with respect to the numbe nction evaluations. We will also present the evolution of the inimal segment difficulty D_{min} (the minimum value of the D's all the segments). This value gives the quality of the solution at certain iteration. In fact, if the function to optimize is bounded D_{min}, we are sure that the best point so far is the global optium.

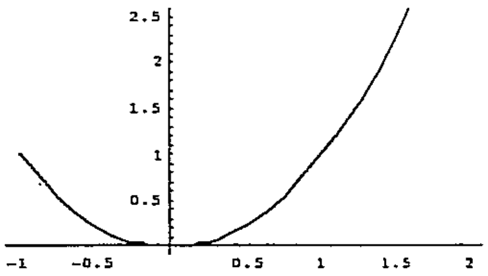
he test functions will be presented by order of difficulty.

. *Parabola*

very method of optimization should be efficient on simple funcons. The first problem presented is to minimize a parabola fined on [−1,2] :

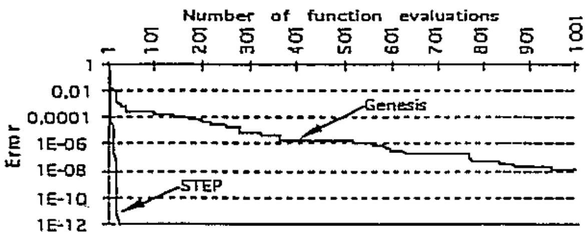
$$f(x) = x^2 \qquad \text{(EQ 18)}$$

GURE 1. Parabola



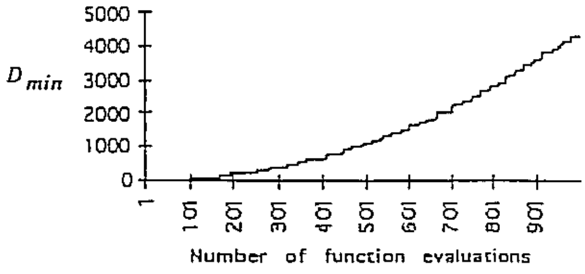
or this function, we compare our algorithm with Genesis only. *lomin* performs a kind of quadratic interpolation to determine its ext try, and finds the optimum in four evaluations.

IGURE 2. Performance for the Parabola



he FIGURE 3. shows the variation of D_{min} according to the umber of function evaluations. Note that D_{min} is roughly proporonal to the square of the number of evaluations.

IGURE 3. D_{min} for the Parabola



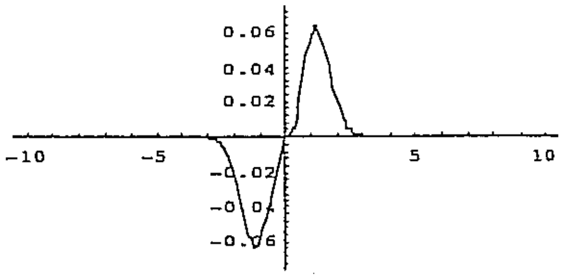
B. *Brent's Fifth Function*

This function is the last one presented in [5] by Brent to demonstrate the performance of his algorithm *Glomin*. It is qualified as difficult by Brent in his book.

$$f(x) = (x - \sin(x)) \cdot e^{-x^2} \qquad \text{(EQ 19)}$$

to minimize over [−10,10].

FIGURE 4. Brent n°5



In order to show the dependency between the performance of *Glomin* and the bound on the second derivative given as an *à priori* condition, we have used the bounds $M = 72$ (noted "Glomin"), and $M = 720$ (noted "Glomin >").

FIGURE 5. Performance for Brent n°5

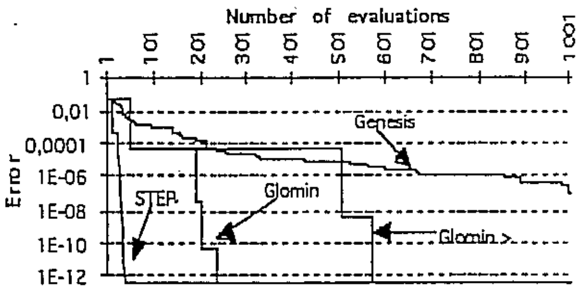
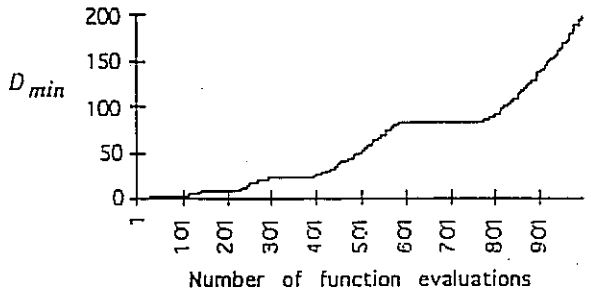


FIGURE 6. D_{min} for Brent n°5



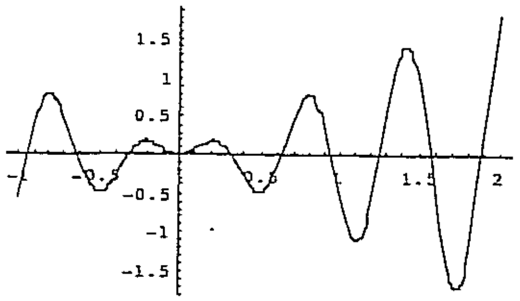
C. *Michalewicz's First Function*

This function is presented in [9] to demonstrate the efficiency of Genetic Algorithms.

$$f(x) = x \cdot \sin(10 \cdot x) \qquad \text{(EQ 20)}$$

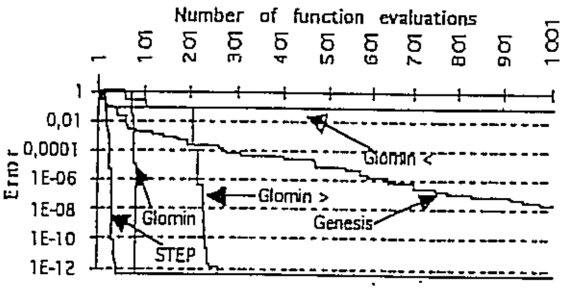
to minimize over [−1,2].

FIGURE 7. Michalewicz n°1



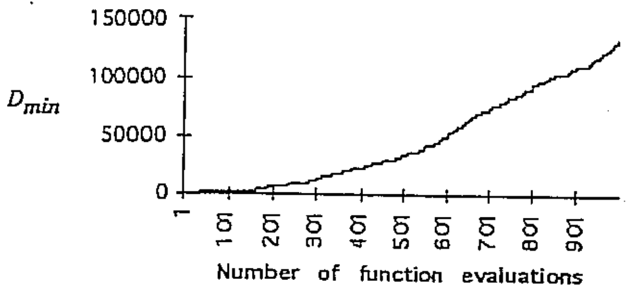
For *Glomin*, we used a bound (over the second derivative) of 100 ("Glomin <"), 400 ("Glomin"), and 40,000 ("Glomin >").

FIGURE 8. Performance for Michalewicz n°1



We can notice the lack of robustness of Glomin if we reduce the bound on the second derivative too much.

FIGURE 9. D_{min} for Michalewicz n°1



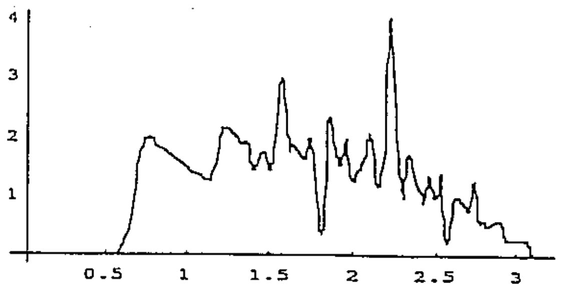
D. *Michalewicz's Second Function*

This function is introduced in [9] to evaluate the performances of certain scaling methods of GA.

$$f(x) = \sum_{i=1}^{10} \sin(x) \cdot \left(\sin\left(\frac{i \cdot x^2}{\pi}\right)\right)^{20} \qquad \text{(EQ 21)}$$

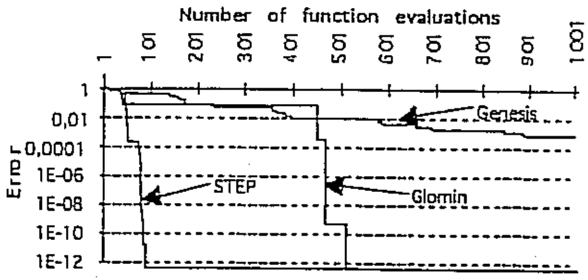
to maximize over $[0, \pi]$.

FIGURE 10. Michalewicz n°2



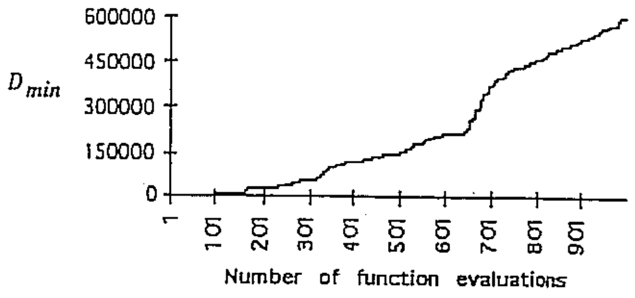
For this function, we have determined by hand an upper bound 650,000 on the second derivative. This value will be used as t parameter of *Glomin*.

FIGURE 11. Performance for Michalewicz n°2



Note that in this case, Genesis seems to be blocked in a local opt mum for about 50% of the experiments.

FIGURE 12. D_{min} for Michalewicz n°2



VI.Conclusions

In this article, we have presented a new, efficient and robus method for optimization in one dimension.

We have seen that in all the cases considered, S.T.E.P. performe better than the other methods. Moreover, S.T.E.P. gives a guarar tee of convergence, in a time comparable to the one needed b *Glomin*. No other method gives such a guarantee.

Besides, while *Glomin* needs *à priori* information about the func tion to optimize (and is very sensitive to a variation of this param eter), S.T.E.P. doesn't require such information, and in additior provides the equivalent data necessary to validate the best poin found as a global optimum after each iteration.